NASA-CR-165,941

NASA Contractor Report 165941

CONTROL SOFTWARE FOR TWO DIMENSIONAL AIRFOIL TESTS USING A SELF-STREAMLINING FLEXIBLE WALLED TRANSONIC TEST SECTION

NASA-CR-165941
19820022438

S. W. D. Wolf

UNIVERSITY OF SOUTHAMPTON
Southampton, England

Grant NSG-7172
July 1982

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

CONTENTS

1. INTRODUCTION

The operating procedure of a self-streamlining wind tunnel has already been discussed [1] and is summarised in the flow diagram shown on Figure 1. The iterative nature of the streamlining process, requiring numerous measurements and calculations coupled with the need for a rapid and continual exchange of data between wind tunnel and computer, makes mandatory the use of a computer.

The current operation of the Transonic Self-Streamlining Wind Tunnel (TSWT) involves on-line data acquisition with automatic wall adjustment. A tunnel run consists of streamlining the walls from known starting contours in iterative steps and acquiring model data. Each run performs what is described as a streamlining cycle. The associated control software is presented here.

Development of the on-line control system for TSWT continues. It is anticipated that modifications to the existing software package will consist of minor changes to some of the subroutines.

The introduction of further subroutines for

(a) Calculation of model forces from the 'wall data' (wall pressures and positions).

(b) Solution of mixed imaginary flowfields with a shock present.

(c) Prediction of wall shapes, to minimise boundary interference on a three dimensional model in a two dimensional test section.

(d) Assessment of residual boundary interferences on a three dimensional model in a two dimensional test section.

is planned as they become available. The modular architecture of the software will allow these additions to be made easily.

The software package has been developed for simple application to other self-streamlining wind tunnels. The modular architecture allows individual control system subroutines to be utilised with existing users software. Also the software package is written in a general manner to minimise new application modification.

During TSWT development numerous programmes have been written to check sections of the control software. A number of these remain in use to assist with TSWT operation as follows:

i) Set both walls to known contours together or individually.

ii) Allow operator modification of known wall contours.

iii) Display current position of both walls.

iv) Display and/or load contents of any specified data file record.

In addition programs have been written to command a Tektronix 4662 plotter to display model pressure distributions, flexible wall Mach number distributions and wall shapes.

2. CONTROL SYSTEM HARDWARE OUTLINE

An overview of the current control system is shown in Figure 2. Macroscopically, the system involves the interaction of the tunnel operator with the wind tunnel and computer to generate the required test data.

The on-line control system has two distinct functions:

1) to streamline the flexible walls which includes assessment of residual boundary interferences.

2) to acquire and reduce test data from a model.

These functions are achieved by two control loops between wind tunnel and computer linked to processing software for data manipulation. The control loops are for wall shape control and pressure data acquisition (Scanivalve control). Data acquisition is automatic since the Scanivalve is stepped by computer commands and analogue pressure data is fed direct to the computer.

The control system hardware has evolved about these two control loops using both analogue and digital data transfer. Microscopically, the control system becomes complex as shown in Figure 3.

The system hardware consists of the computer and its peripherals communicating with a series of control and signal conditioning sub-systems housed in a control cabinet by the wind tunnel. This cabinet is then connected to the test section wall position sensors (linear potentiometers) and wall jack stepper motors[2].

The hardware will perform four functions:

1) wall movement.

2) wall and model pressure measurements.

3) wall position sensing.

4) system monitoring (not yet existing).

The wall movement function involves the loading of forty motor latch boards with direction information (stop, forward-go or reverse-go). Then the sending of a 'go' pulse to the Pulse Sequence Generator (PSG), starts actual wall movement. The PSG generates control pulses at a fixed frequency to the forty Motor Drive Boards (MDBs). The MDB, using signals from the motor latch board and the power

supply, controls the sequence of power pulses transmitted to the 3-phase stepper motor powering each wall jack. After a pre-determined and variable time interval the PSG is switched off and wall movement ceases. The wall movement control sequence is then repeated until satisfactory wall contours are achieved.

The wall and model pressure measurements involve the driving of the scanivalve system by step pulses, to a required sampling port. Then the analogue signals from the pressure transducers are sampled by the computer after a suitable settling time (i.e. 50 milli-seconds).

The wall position sensing function involves the sampling of forty analogue signals from the linear potentiometers attached to each wall jack mechanism. These signals are continuously available at the computer peripherals, but they have been found to be susceptible to electronic interference when the jack motors are switched on. Hence the wall position is only sampled with the walls stationary.

The system monitor is a necessary part of a practical digital control system. When this hardware becomes available it will provide information on the status of hardware components, to allow quick error diagnostics during tunnel operations.

The conditioned analogue signals from the test section (within the range ± 5 volt) are fed via the control cabinet to a DEC AD11-K Module for 12-bit analogue to digital coversion. This module is combined with a DEC AM11-K Multiplexer to make 64 channels available for input signals.

The digital signals, at 0 and 5 volts , are transmitted to the wind tunnel in a code described in Appendix A. Device selection is by means of a 'telephone exchange' called the "address decoder". In fact, commands to the test section are sent to all devices but only one device is enabled to read the information, by address decoder selection. The versatility of a digital control system is well known and the reduction in interface wiring compared with an analogue system is significant. The address decoder has the capability of addressing 64 devices.

The control system hardware layout in Figure 3 has been simplified for clarity. In practice there are numerous synchronisation signal paths between devices, to ensure correct operation sequencing and to prevent 'race' problems.

The tunnel operator monitors the control system from a computer VDU consol and inputs test parameters. The consol allows the real time display of test section and model data which is stored on the computer mass storage device. This data can subsequently be drawn in graphical form on an XY Plotter (Tekronix 4662).

Hard copy of summarised run data is printed in real time on a line printer. The control of a model wake traverse mechanism has been incorporated in the system, utilising the wall movement technique already described.

The control system hardware has excess capability with 11 spare digital address slots and 15 spare analogue input channels. This spare capacity may be used for the control of tunnel Mach number and model attitude in the future.

3. CONTROL SOFTWARE

Computer software has been developed for the on-line TSWT control system using a versatile modular architecture. Hence the program has been reduced to a collection of manageable subprograms which can be combined to control the wind tunnel and output real time results or provide more detailed off-line re-analysis of previously acquired data.

An overview of the control software package is shown below.

File Type & Name	File Storage Name	Function
Main Program (TSWT)	OFLEX	i) Control and sequence subroutine calls. ii) Read test parameters from the operator.
Subroutine 1 (DATA)	OAD	Acquire pressure data from the wind tunnel.
Subroutine 2 (REDUCE)	ODR	Read tunnel data from disc storage and reduce raw pressure data from the wind tunnel.
Subroutine 3 (WAS)	OJUDD	Perform wall setting calculations.
Subroutine 4 (STAR)	ODST	Calculate local boundary layer displacement thickness and Mach number along each wall.
Subroutine 5 (SUME)	OERR	Assess wall induced interferences at the model.
Subroutine 6 (FORCE)	OWING ONPL	Calulcation of model forces for NACA 0012-64 and NPL 9510 sections respectively.
Subroutine 7 (SET)	OUT	i) Store run data on disc ii) Output data to the terminal and/or the plotter.
Subroutine 8 (WALL)	OADJ	Move the walls to new contours

This breakdown of the software into modules has been extremely useful, particularly for storage, editing and debugging purposes.

The software written in FORTRAN IV language is run on a DEC 11/34 with a DEC RT-11 V4 operating system. The software is linked to a system library and a FORTRAN library to access functions and system subroutines and a Real Time System Library (RTSL) to access peripheral control subroutines. The complete compiled and linked program requires over 100 blocks (25.6K words) of memory spare.

Current 16-bit computer processors are only capable of addressing 32K words (64K bytes) of real memory space. But of this, only 22K words is available for a user's program, depending on the size of the operating system. Therefore to run the TSWT control software on a 16-bit machine a technique of overlaying has to be used, so that only part of the software is stored in the real memory at any instant during execution.

Each subroutine is a self contained program communicating with the main program via common data blocks. So in theory only one subroutine is required in the real memory at any one time for execution. In practice, the subroutines have been grouped together to minimise the number of overlays, thereby reducing the time required for overlaying. The overlaying structure of the control software is shown below

	SEGMENT 1	SEGMENT 2	
OVERLAY REGION	Subroutine 1 Subroutine 2 Subroutine 3 7762 words	Subroutine 5 Subroutine 6 Subroutine 7 Subroutine 8 6558 words	7762 words
ROOT SEGMENT	Main Program +	System Library FORTRAN Library RTSL Library	9470 words

This program structure is implemented at 'link' time during the program generation cycle as below

```
R LINK
*OFLEX = OFLEX,FORLIB,RTSL/C
*OAD,ODR,OJUDD/O/:1/C
*OERR,OWPL,OUT,OADJ/O:1
```

These commands generated a runable program called OFLEX. The control software memory requirement was reduced from 27k words to 17.3k words.

At 'run' time the program OFLEX requires four data files to exist on the computer mass storage device. Data file ADC.DAT receives the raw analogue-to-digital counts of the 'wall and model data' for each streamlining iteration; PAD.DAT provides and receives sets of wall contours and the associated external imaginary wall velocity distributions; NPL.DAT or WING.DAT receives Cps from the NPL 9510 and NACA 0012-64 models respectively, for each streamlining iteration; TWST.DAT holds all fixed tunnel data, i.e. jack positions, potentiometer calibrations, scaling and coupling factors, matrix coefficients for camber interference assessment and boundary layer information. RUN.DAT holds run data, i.e. ambient temperature and pressure, run number, iteration record number, and number of model tappings. The data files ADC.DAT, PAD.DAT and NPL.DAT/WING.DAT each hold 50 records: Records 1 to 3 in PAD.DAT hold data on the three aerodynamically straight contours (i.e. position and boundary layer thickness). Records 1 and 2 of NPL.DAT and WING.DAT files are used to store model tapping X and Y co-ordinates relative to the model leading edge. Records 4 to 50 are available to store data from each streamlining iteration. Hence iteration record numbers range from 4 to 50. When the iteration record number equals 50, ADC.DAT, PAD.DAT and NPL.DAT/WING.DAT must be copied to a data bank since the original data is then overwritten by subsequent iterations, starting with record 4. The upper limit on the iteration record number has been chosen to keep the data files in manageable proportions (i.e. 25.6k words maximum size). The total storage requirement of data files to run OFLEX in 61.7k words.

The data file RUN.DAT must be loaded with current run data before the control software is activated. This operation is performed by running a program OSTART, the software of which, called RUN,is described in section 3.11.

A complete listing of the control software is described in the following sections. Where possible standard FORTRAN has been used but peripheral control commands are peculiar to the DEC system used. These subroutine calls can be grouped into Analogue to Digital sampling commands (ADC and RTS), and programmable clock commands (SETR and LWAIT). In addition there are calls to the system library routines (IPEEK and IPOKE) for digital input and output operations.

An example of the minimal printout from OFLEX associated with a typical streamlining cycle, involving two wall adjustments, is shown on Figure 5. The walls were initially set to contours stored in record 30 of file PAD.DAT and the run finished with wall contours as stored in record 32. The model had no pressure tappings and therefore no wing performance data was presented in the print-out. Note that the print-out from OFLEX is sent to a line printer (logical Unit 7) while operator information is sent to a VDU Consol (Logical Unit 5). The operator information consits of prompts to indicate the stages reached in the streamlining cycle and error warnings.

The versatility of the software has allowed simple generation of programs for particular tasks such as tunnel data re-analysis. Using the existing subroutines as building blocks, a new program has been made up of a series of these subroutines linked to a new main program. For example, data re-analysis is achieved by running the program ORLEX. The main program REAN is a modification of TSWT with different subroutine calls and an extended print-out demanded as described in section 3.10. The program structure is very similar to that of the control software and is generated using the following link command with a memory requirement of 17.1k words.

```
R LINK
*ORLEX = OREF,FORLIB,RTSL/C
*OAD,ODR,OJUDD/O:1/C
*OERR,ODST,OUT/O:1/C
```

where OREF is the file storage name of program REAN

An example of the extended printout is shown on Figure 6 for a typical re-analysis of raw TSWT data for one iteration of run 389.

Should any new analysis technique become available, then a new subroutine could replace or supplement the existing subroutines, and be incorporated in OFLEX and/or ORLEX by minor adjustments to the main program and the LINK commands.

3.1. Program TSWT

The main control program listed on Figure 4.1 reads tests parameters from the tunnel operator and sequences subroutine calls:-

Lines 0002 - 0009				Define all common data blocks used thus:-
Block 1		NJ	=	Effective number of jacks per wall (real plus dummy).
		MT	=	Number of model pressure tappings.
		NR	=	Number of tunnel reference samples during a scanivalve scan.
		CL1	=	Printout control value.
		B1	=	Prandtl-Glauert scaling factor = $\sqrt{1-M^2}$
		PR1	=	For development only.
		AK1	=	Aerodynamic coupling factor.
		AK3	=	Wall movement scaling factor.
		AN	=	Model angle of attack (degrees).
		R3	=	Chord Reynold s number.
		PP2	=	Dynamic pressure allowing for compressibility.
		ITRN	=	Run number.
Block 2	Array	RN	=	Bottom wall potentiometer outputs, volts.
	"	RS	=	Top wall potentiometer outputs, volts.
	"	W	=	Bottom wall imaginary wall velocities.
	"	X	=	Top wall imaginary wall velocities.
	"	P	=	Top wall real wall static pressures.
	"	C1	=	Bottom wall real wall static pressures.
	"	RD	=	Development data.

Block 3	Array	D	=	Measuring point co-ordinates.
	"	WTY	=	Top wall movement from the straight wall contours (Inches).
	"	WBY	=	Bottom wall movement from the straight wall contours.
	"	WL	=	Effective panel length per wall tapping.
	"	B	=	Model Cps.
	"	PD	=	Transducer Cps.
Block 4	Array	E	=	Calculated imaginary external top wall velocities.
	"	H	=	Calculated imaginary external bottom wall velocities.
	"	Y	=	Predicted top wall movements required (Inches).
	"	G	=	Predicted bottom wall movements required (Inches).
	"	WI	=	Real top wall velocities squared $(V/U_\infty)^2$
	"	XI	=	Real bottom wall velocities squared $(V/U_\infty)^2$
Block 5	Array	U	=	Difference between real and imaginary top wall velocities (u/U_∞)
	"	V	=	Difference between real and imaginary bottom wall velocities (u/U_∞)
	"	DS	=	'Straight' wall local δ^* values.
	"	DET	=	Local $\Delta\delta^*$ values.
	"	CS	=	Matrix coefficients for camber interference assessment.

Block 7	Array	PC	=	Potentiometer calibrations (Volts/Inch)
Block 8		TAB	=	Ambient Temperature (Deg. c)
		AMP	=	Ambient Pressure (Cm Hg)
		IFN	=	Iteration Record No:
		IAM	=	Automatic In-file selection trigger (O-off; 1-on)
Block 9	Array	IX	=	Pressure channel zeroes.

Line 0010	Initialise Arrays X & DET.
Line 0012	Set streamlining trigger (PR1) to zero (i.e. walls not streamlined).
Line 0013	Set automatic in-file selection trigger (IAM) to zero (i.e. Non automatic selection).
Lines 0014 - 0015	Define File 5 as data file RUN.DAT.
Lines 0016 - 0022	Read run data from RUN.DAT and load ITRN with the run number, IFN with iteration record number, TAB with ambient temperature, AMP with ambient pressure and MT with number of model taps.
Line 0023	Set extended printout trigger (CL1) to zero (i.e. minimal print-out required).
Line 0024	NR = Number of tunnel reference samples in a pressure scan.
Line 0025	NJ = Number of measuring points per wall (See Figure 7).
Line 0026 - 0048	Input test parameter: Angle of attack (AN)
Lines 0049 - 0059	Subroutine call sequence.
Line 0053	If no model tappings (i.e. MT=0) do not call Subroutine FORCE.
Line 0057	If walls streamlined (i.e. PR1=1) do not call Subroutine WALL

Lines 0061 - 0064	When the Iteration Record number is 50 and jump out of streamlining loop.
Line 0066	Increment the Iteration Record number
Line 0067	If walls are un-steamlined (i.e. PR1=0) continue the streamlining cycle.
Lines 0069 - 0078	Load run data array TUN and load file RUN.DAT with new values of TUN.

3.2. Subroutine DATA (Incorporating Subroutines GAD and STEP)

This subroutine primarily acquires all the tunnel pressures either from the tunnel via an automatically stepped scanivalve or from disc storage. Some data is then reduced.

The option to acquire old data stored on disc allows re-analysis of previous runs. All analogue to digital conversions associated with the pressure data are handled by software in Subroutine GAD. Automatic control of the Scanivalve is achieved by software in Subroutine STEP. The listing on Figure 4.2 can be broken down thus

Lines 0002 - 0003	Dimension of data blocks DA, DB, DC and IDATA.
Lines 0004 - 00011	Define only common data blocks required in Subroutine DATA.
Line 0012	Initialise the first row of 2D- array IDATA.
Line 0013	CHD = Model chord = 4 inches (NACA 0012-64) = 6 inches (NPL 9510)
Line 0014	NJ1 = number of wall jacks.
Line 0016	NR = number of tunnel reference samples in a pressure scan.
Line 0017	IP1 = Approximate low value of scanivalve position encoder output at pressure scan start condition.
Lines 0018 - 0021	Transducer calibration valves (cm Hg per A-D count).

Lines 0025 - 0032	If a re-analysis (i.e. CL1=500), open data file 4, select an existing record (IR) and fill IDATA with raw transducer A-D counts from a previous TSWT run.
Lines 0034 - 0037	Load pressure channel zeroes into IDATA
Lines 0038 - 0040	Identify all passes of the data acqination cycle other than the first and print a prompt.
Line 0041	Initiate DO loop to acquire tunnel pressures using the scanivalve system.
Lines 0044 - 0051	If the first data acquisition cycle (i.e. X(30)=0) read the four pressure transducer zero signals and halt the software until a 'CR' is received from the terminal - transmitted when the tunnel flow is stabilised by the operator.
Lines 0053 - 0061	Wait for a change in the encoder output generated by a scanivalve step using Subroutine STEP. If there is a step error halt program.
Line 0062	Sample four transducer outputs using Subroutine GAD and load array IDATA.
Lines 0064 - 0067	Load pressure channel zeroes into array IX
Lines 0068 - 0073	Open a data file 4 and fill a selected record (IOT) with 192 raw A-D count values from the iteration, stored in IDATA.
Lines 0074 - 0080	Convert raw data into pressures relative to tunnel reference pressure (cm Hg), and load into array PD.
Lines 0081 - 0093	Load arrays P and Q with wall jack centreline pressures with dummy measuring points included (See Figure 7).
Lines 0094 - 0101	Load arrays P and Q with mid-jack centreline pressures.
Line 0102	RD (NR + 1) = test section total pressure reading.
Line 0103	RD (NR + 2) = Ambient pressure (cm Hg).
Line 0104	RD (NR + 3) = Ambient temperature (deg C).
Line 0106	R1 = Stagnation pressure (cm Hg)

Lines 0107 - 0123	Calculate tunnel reference Mach numbers at regular intervals through the pressure scan, and look for excessive fluctuations (i.e. $\Delta M > .01$).
Lines 0124 - 0140	Calculate selected tunnel pressures for possible calibration checks.
Line 0141	Store average tunnel reference static pressure in B1.

Subroutine GAD

Line 0001	Subroutine label with data transfer of value N and array IDATA.
Line 0002	Dimension of IDATA (Average A-D counts) and IDT (Raw A-D counts) arrays.
Line 0003	NS = Number of samples per channel
Lines 0004 - 0006	Time delay of 50 milli-seconds using the programmable clock (Subroutine SETR).
Lines 0007 - 0011	Take NS samples of four analogue input channels at 1kHz and store digital results in IDT.
Lines 0012 - 0020	Convert all raw A-D counts from packed integer format to real number format (Subroutine CVSWG) then take the average of the NS samples per channel and store in IDATA.
Line 0021	Stop the programmable clock.

Subroutine STEP

Line 0002	POKE digital output register with the command number 16384 (See Appendix A). Note no scanivalve address is required with the present hardware configuration.
Lines 0003 - 0005	Time delay of 20 milli-seconds using the programmable clock.
Line 0006	POKE digital output register with a zero value to clear.

3.3. Subroutine REDUCE

This subroutine performs data reduction operations on tunnel pressure information and acquires further tunnel data from data files held on the mass storage device. The software listed on Figure 4.3. is described thus:

Lines 0004 - 0010	Define only common data blocks required in Subroutine REDUCE.
Lines 0011 - 0015	Set variables, as previously described for subroutine DATA.
Lines 0016 - 0025	Calculate the average freestream Mach number of the run.
Line 0026	PP2 = qI/q_c compressibility correction to tunnel q.
Lines 0027 - 0041	Calculate chord Reynold's number (R3).
Lines 0042 - 0054	Convert tunnel pressure data into Cp using the tunnel reference pressure associated with each group of six readings.
Lines 0055 - 0080	Load model pressure coefficients into array B from transducer channels 1 and 3 data. Note this software is model dependent and configured for use of the NPL 9510 section.
Lines 0083 - 0092	Specifiy IWF corresponding to 'Straight Wall" base data for Mach number panels M < .725, .725 <M> .825 or M > .825 for re-analysis case only. Otherwise a dummy data base is used (i.e. IWF=1).
Lines 0094 - 0096	Load arrays DA, DB and PC with data from File 2 (TUN.DAT).
Lines 0097 - 0098	Define File 3 as data file PAD.DAT.
Lines 0099 - 0109	Define wall contour record number automatically (i.e. IAM=1) or by operator input.
Line 0110	Load array DC with 'wall data' from File 3.

Lines 0111 - 0118	Load arrays WTY, WBY and DS with data from array DA.
Lines 0119 - 0124	Load arrays CS and D with data from array DB
Line 0125	Read coupling factor (AK1) and scaling factor (AK3) from File 2, record 10.
Lines 0126 - 0129	Calculate effective wall lengths between mid-jack wall points and store results in array WL.
Lines 0130 - 0139	Load arrays RS, RN, X, W with data from array DC (See section 3.1 for array descriptions).
Lines 0140 - 0145	Convert the wall Y co-ordinates into movement from straight contours in inches.
Lines 0146 - 0147	Close data files
Line 0148	Signify the completion of the data acquisition cycle by setting X(30) = 1.

3.4. Subroutine WAS

The wall adjustment strategy[1,2,3] (WAS) manipulates the calculated imbalance between real and imaginary wall velocities to generate a wall movement which will give zero wall loading or vorticity.

The strategy requires interpolation of real wall velocities at regular intervals along each wall. To ensure accurate interpolation at the wall ends using curve fitting to the wall velocity distribution, two straight dummy wall extensions are added to each flexible wall, 15.25 cm (6 inches) and 22.86 cm (9 inches) in length respectively. The dummy wall measuring points introduced are points 1 and 2 upstream where the wall velocity is assumed free-stream and points 22, 23 and 24 downstream, which have a wall velocity equal to that of measuring point 21 or jack 19. Measuring point 1 is also assumed to be the origin of the wall boundary layer. Measuring points 3 to 21 are real and correspond to the position of wall jacks. The software representation of each flexible wall, in terms of measuring point is shown in Figure 7. The software listed on Figure 4.4. is described below

Lines 0002 - 0006	Define only common data required in subroutine.

Line 0007	Dimension working arrays.
Lines 0009 - 0010	Message to tunnel operator.
Lines 0011 - 0012	Equalising top and bottom wall coupling and scaling factors.
Lines 0016 - 0038	Compute external velocities for next wall shape where

Q1	=	compressible dynamic pressure
TEMP1 & TEP1	=	incompressible top and botton wall Cps.
U & V	=	inbalance between real and imaginary wall velocities (Top and bottom walls).
E & H	=	external velocity perturbations computed for the next top and bottom wall shapes (v/u_∞)
XI & WI	=	real wall velocities squared $(v/u_\infty)^2$

Lines 0040 - 0041	Load Z with interjack streamwise spacing co-ordinates.
Lines 0042 - 0044	Set up DO loop for both walls.
Lines 0045 - 0053	Load A & XB with D & U or V data in sets of four values for top or bottom wall.
Lines 0054 - 0067	Cubic spline fit to each set of data to obtain wall velocity imbalance between jacks.
Lines 0069 - 0098	Summation of velocity induced by the vorticity distributed along a wall at each jacking point. The result is stored in arrays S for the top wall and T for the botton wall.
Line 0099	End of wall velocity analysis repeated for each wall.
Lines 0104 - 0137	Numerical integration of top and bottom wall normal velocity components to generate jack movement demands. Raw movements are stored in TT and R. Arrays Y & G store the effective jack movements after scaling. Likewise E and H (the external velocity perturbations) are modified by scaling in lines 0133 & 0134. In lines 0135 & 0136 values in arrays Y and G are converted to compressible values using $B1 = \sqrt{1-M^2}$.

3.5. Subroutine STAR

The function of STAR is to calculate Mach number and boundary layer displacement thickness values for each wall jacking point necessary only for reanalysis information. The boundary layer calculations use a numerical solution of the Von Karman Momentum Integral equation for a turbulent boundary layer. The software listing is shown on Figure 4.5

Line 0002	Dimension of array DELTA for δ^* storage.
Lines 0003 - 0007	Define only required common data blocks.
Lines 0008 - 0011	Calculation of isentropic flow relationships.
Lines 0012 - 0013	Calculation of air density (kg/ft^3)
Lines 0014 - 0015	Calculation of air temperature (K).
Lines 0016 - 0017	Calculation of air viscosity (lb/ft_{sec}).
Line 0018	CL1 alue selects extended printout when equal to 500
Lines 0027 - 0145	Overall Do loop performs calculations for each jack on both walls using sets of three jacks, labelled 0, 1 and 2.
Lines 0044 - 0045	Load X1 and X2 with jack 1 and 2 co-ordinates relative to jack 0.
Lines 0046 - 0051	Ensure that the correct tunnel reference pressure is used with each wall pressure.
Lines 0053 - 0062	Load SP1, SP2 and SP3 with top or bottom wall velocity perturbations.
Lines 0063 - 0068	Calculate wall velocities U0, U1 and U2 (ft/sec)
Lines 0069 - 0077	Calculate local wall mach number and store in P (top wall) or Q (bottom wall).
Lines 0081 - 0087	For the second measuring point assume a turbulent boundary layer growth according to $\delta^* = \dfrac{.0213x}{R_x^{0.1428}}$
Lines 0088 - 0092	Calculate the velocity gradient at jack 1 and store value in D2.

Lines 0093 - 0094	Guess the size of δ^* and store as P1.
Lines 0096 - 0099	Calculate components of M.I. equation.
Lines 0100 - 0105	Check if the measuring point is a special case (points 3 or 24 along each wall) and calculate S1 with appropriate velocity gradient (S1 = rate of change of δ^* with streamwise distance).
Lines 0106 - 0111	Calculate δ^* from $d\delta^*/_{dx}$ and store value in P4.
Lines 0114 - 0122	Home-in on correct value of P1 (If P4 > P1 increase P1 by 25 E-6).
Line 0123	Load array DELTA with δ^* values (inches).
Line 0127	Calculate $\Delta\delta^*$ from 'straight wall' to contoured wall.
Lines 0131 - 0135	Output results to a terminal if CL1=500.
Lines 0136 - 0141	Isolation of special cases requiring P4, ST & D2 results to be stored in new addresses.
Line 0142	Load array DELTA with δ^* values (inches).
Lines 0146 - 0163	Calculate wall Mach numbers between jacks and store results in arrays P and Q, corresponding to top and bottom wall data.

3.6. Subroutine SUME

SUME assesses the residual interferences, due to any residual wall loading, along the tunnel centre-line and the model chord line. The three components of the interferences are related to estimated changes in C_L. The quality of streamlining is then assessed from these interferences and a measure of the average pressure imbalance across the walls. A listing is shown on Figure 4.6.

Lines 0002 - 0007	Define only required Common Data blocks.
Lines 0008	Define model in use IWT = 0 for NACA 0012-64 IWT = 1 for NPL 9510
Line 0010	CHD = Model chord (inches)
Line 0013	OR = X co-ordinate of model pivot point which is at ¾ chord and nominally mid way between the straight walls (Inches).

Lines 0014 - 0016	Calculate model chord line X and Y increments and store in AX and AH respectively.
Lines 0017 - 0018	IPP = Position of model pivot point as a number of $\frac{1}{8}$ chords.
Line 0020	OR = X co-ordinate of the model $\frac{1}{4}$ chord point - the origin.
Line 0021	Subroutine print trigger (OT) set to zero for print-out.
Line 0023	Eliminate dummy jacks by reducing number of jacks per wall by four.
Line 0025	Y3 = nominal tunnel semi-height (Inches).
Lines 0030 - 0037	Calculate imaginary wall velocity squared and sum the absolute value of the load difference between real and imaginary velocities squared for each jack. Store the results in EE (top wall) and F (bottom wall).
Lines 0038 - 0043	Calculate the average C_p difference between real and imaginary flows for each wall (called E) and store as ET and CEB, and print-out.
Line 0046	If minimal print-out is specified (i.e. CL1 = 0) jump to line 0083
Line 0050	For 37 points along the tunnel centreline summate the effect of wall voriticity, at the model and along the tunnel centreline.
Line 0051	X1 = X co-ordinate of measuring point.
Lines 0054 - 0074	Sum the velocity components of the wall vorticity at co-ordinate X1 on the tunnel centreline and store results in arrays UT and VT.
Line 0055	X2 - Horizontal displacement between jack and analysis point (Inches).
Line 0057	Y2 = Displacement of bottom wall downwards from the centreline (Inches).

Line 0065 Calculate the horizontal induced velocity perturbation component due to vorticity at measuring points n. (See Figure 7)

$$U1 = \frac{Z_n}{2\pi} \times \sum_{Top}^{Bottom} \frac{Y_n}{X_n^2+Y_n^2} * \Gamma_n$$

where Γ_n = Vortex strength

= Local imbalance of wall real and imaginary velocities.

X_n = horizontal separation of analysis point and a vortex assumed at a measuring point n.

Y_n = vertical separation of same.

and Z_n = panel length = distance between mid-jack points spanning measuring point n.

In software notation

$$U1 = \frac{WL}{TOPI} *((U * R2) + (V * R3))$$

where WL, U and V are arrays.

Lines 0067 Calculate the vertical component of induced velocity perturbation V1 due to vorticity at measuring points n

$$V1 = \frac{Z_n}{2\pi} * \sum_{Top}^{Bottom} \frac{X_n}{X_n^2+Y_n^2} * \Gamma_n$$

Lines 0076 - 0079 If OT = 0 print-out results.

Lines 0083 - 0119 Calculate velocity perturbations for nine equidistant points along the model chord line.

Line 0091 Calculate induced flow angles along the model chord line (AA2).

Line 0092	Calculate induced C_p (CP1).
Line 0093	Store ¼ chord point C_p in address CP.
Line 0103 - 0105	Load A1 and A2 with leading edge and trailing edge induced flow angles.
Line 0107	SP = Contribution to induced camber.
Lines 0110 - 0112	Differentiate between odd and even numbered measuring points.
Lines 0114 - 0118	Load accumulators for Simpsons rule numerical integration technique.
Line 0120	CPE = Average C_p errors induced along model chord.
Line 0123	P3 = Induced camber in terms of ΔC_L.
Line 0124	A3 = Induced camber (degrees).
Line 0128	CL = Induced Angle of Attack in terms of ΔC_L.
Line 0129	A1 = Induced Angle of attack (degrees).
Lines 0130 - 0139	If a re-analysis print-out all residual interference information (i.e. CL1 = 500).
Lines 0140 - 0142	Print abbreviated version of residual interference information if CL1 = 0.
Lines 0145 - 0154	Increment ISC for each quality of streamlining satisfied.
Line 0156	If all the qualities of streamlining are satisfied (i.e.ISC = 5) PR1 = 1, the walls are streamlined.
Line 0160 - 0163	If the walls are streamlined print messages to the operator and the line printer.

3.7. Subroutine FORCE (Based on a NASA numerical technique)

FORCE integrates the model pressure distribution to give coefficients C_L, C_D and C_M.

3.7.1. NACA 0012-64 version - listed on Figure 4.7a.

Lines 0002 - 0006	Define required Common Data blocks.
Lines 0007	Dimension working arrays.
Line 0010	MT = Number of model tappings +2 for dummy leading edge tappings.
Lines 0011 - 0012	Open data file 2 (WING.DAT)
Line 0013	Load array AG with X co-ordinates of model tappings (upper surface followed by lower surface).
Line 0014	Load AJ with Y co-ordinates as for the X co-ordinates.
Line 0015	IHT = Number of model tappings per surface.
Lines 0016 - 0023	Titling and test parameter input from the terminal (AN = angle of attack - degrees).
Lines 0024 - 0029	Calculate an effective vertical component for each model tapping, and store in array AH.
Lines 0031 - 0040	Initiate a DO loop with special processing for first and last tappings on each surface and different anlysis for each surface.
Line 0042	WF = Weighting Factor, in this case for C_N.
Line 0043	L6 = Local components of C_N for upper surface.
Line 0044	L5 = Sum of C_N components.
Line 0045	Store negative L6 in array A.
Lines 0047 - 0050	Calculate the upper surface C_N component for the dummy leading edge tap.
Lines 0051 - 0056	Calculate the C_N component for the last downstream tap on the upper surface.
Line 0055	C_N = Upper surface C_N.
Lines 0057 - 0061	Calculate the lower surface C_N component for the dummy leading edge tap.
Lines 0062 - 0067	Calculate C_N component for the last downstream tap on the lower surface.

Line 0066	CNL = Lower surface C_N.
	L5 = Total model C_N.
Lines 0068 - 0072	Calculate C_N components for lower surface tappings.
Lines 0074 - 0088	Do loop to calculate C_L components for each tapping.
Line 0077	C5 = Upper surface C_c component.
Line 0078	Store C5 in array AH.
Line 0079	C3 = Sum of C_c components.
Line 0083	CCP = Suuper surface C_c.
Line 0085	C5 = Lower surface C_c component.
Line 0089	St = Lowers surface model C_c.
	C3 = Total C_c.
Lines 0098 - 0122	Do loop to calculate C_m components about the model leading edge.
Line 0103	BB = Upper surface C_m components.
Line 0104	C9 = Sum of upper surface C_m components.
Line 0105	TIT = Tapping position (% chord).
Line 0115	BB = Lower surface C_m components.
Line 0116	C8 = Sum of lower surface C_m components.
Line 0123	ST1 = Total C_m.
Lines 0124 - 0131	Output forces C_N, C_c, C_m.
Lines 0132 - 0136	Calculate coefficients C_L and C_D from C_N, C_c and angle of attack.
Line 0137 - 0140	Output wing performance coefficients.
Line 0141	Close data file.

3.7.2. NPL 9510 version - listed on Figure 4.7b

This version of FORCE is the same as the NACA 0012-64 version in all respects, except for the number of model tappings (50 in total, with a split 32:18 on upper and lower model surfaces respectively) and the labelling associated with the print-

out. Also notice that print-out control commands ahve been inserted at lines 0015 , 0092, 0108, 0112, 0120 and 0126. Setting CL1 = 0 suppresses all print-out except data associated with the label 'WING PERFORMANCE'. NPL 9510 model Cps are stored in NPL.DAT for subsequent plotting (See next section).

3.8. Subroutine SET

SET controls the output of tunnel data to the VDU consol and the line printer and the output of results to disc storage. This software is configured for testing the NPL 9510 section. A listing is shown on Figure 4.8.

Line 0002	Dimension data array DA.
Line 0003 - 0009	Define required Common Data blocks.
Line 0010	Initialise some array elements for print-out purposes.
Lines 0011 - 0012	Open wall output file 3 with 50 records each 512 words in length (PAD.DAT).
Lines 0013 - 0014	Open wing output file 2 with 50 records each 128 words in length (NPL.DAT).
Line 0015	NJ1 = Number of wall jacks.
Line 0016	Define model used: IWT = 0 for NACA 0012-64; IWT = 1 for NPL 9510.
Line 0017	CL1 = 0 for minimal print-out, so jump to line 0055
Lines 0019 - 0035	Output of pressure transducer Cps, external wall velocity for each jack and the predicted jack movement for zero wall interference.
Lines 0036 - 0039	Print-out top wall labels.
Lines 0042 - 0043	Define File 6 as WALM.DAT
Lines 0044 - 0053	Output wall Mach number distributions stored in arrays P and θ into an operator set (IMR) record of File 6 for subsequent plotting.
Lines 0055 - 0080	Store potentiometer values for new wall contours in array DA. If CL1 = 500 print-out information on jack X & Y co-ordinates, local Mach number, current potentiometer volt values, predicted potentiometer volt values for the next iteration.

Line 0060	WM = Effective position of top wall contour allowing for boundary layer growth along the wall.
Lines 0061 - 0069	If movement demands for the downstream jacks (No. 16-20) exceed mechanical limits in terms of pot volts, set to safe minimum values.
Line 0078	If CL1 = 500 print out the local wall Mach Number between jacks.
Lines 0081 - 0115	Repeat analysis and print-out for the bottom wall.
Lines 0105 - 0114	Calculation of wall Mach number standard deviation for top and bottom walls for use in special re-analysis of tests with an empty test section.
Line 0116	Set KT to equal the total number of jacks.
Lines 0117 - 0118	Store top wall external velocity perturbations in array DA.
Lines 0122 - 0126	Print prompts on wall and model output records.
Line 0128	Write array DA into record IFN of File 3 on disc.
Lines 0129 - 0132	Define the number of model tappings and store model Cps in record IFN of File 2 on disc.
Lines 0133 - 0134	Close data files 2 and 3.

3.9. Subroutine WALL (incorporating Subroutines INIT, START, MOVE and DIO)

This subroutine controls the test section wall adjustments. Both walls are moved simultaneously in variable increments of movement. Each jack is commanded to move away from the model in its last adjustment to ensure the walls are rigid to air pressure loads. Numerous saftey checks are included to guard against jacks jamming or jacks out of control.

Subroutine INIT initialises the control system. Subroutine START switches the jacks on and off for a time interval proportional to the average position error of the wall adjacent to the model pressure surface. Subroutine MOVE loads each jack with direction information and checks the data is loaded. Subroutine DIO handles all the Digital input and output operations between wind tunnel and computer.

The software is dependant on the tunnel hardware and is listed on Figure 4.9

and described thus

Line 0002	Define Common data required.
Line 0003	NWJ = total number of wall jacks.
Lines 0004 - 0005	Define arrays thus PC = potentiometer calibration (volts per inch) IM = stores jack movement demands for checks. ADC = stores wall pot. volts for movement checks. PV = wall contour pot. volts. PL = minumum safe pot. volt readings for downstream jacks.
Line 0006	Load array PL with values
Lines 0007 - 0010	Define File 3 as PAD.DAT and load array PV with a set of wall contour pot. volts from record IFN of File 3.
Line 0011	IF = 1 for the first pass through the wall adjustment cycle
Line 0012	IP = anti-backlash overshoot increment.
Line 0013	Initialise the control system.
Lines 0014 - 0016	Load variables IW = Wall number (0:top/1:bottom) ICNT = No. of attempts to move one jack. IS = No. of jacks correctly positioned.
Line 0017	ITOL = Wall setting tolerance band in pot. volts.
Line 0018 - 70	Calculate movement required for each jack from its current location and load each jack with direction information.
Lines 0019 - 0021	Read the potentiometer output for jack (L), convert the result to pot. volts, and store in IPV.
Line 0022	IMOVE = Actual jack movement from previous position. Ignored on first pass.
Line 0023	Store current jack pot. volts (IPV) in array ADC.
Lines 0024 - 0027	If IPV > 980 (i.e. jack approaching mechanical limit) then stop.

Lines 0029 - 0036 — Single out downstream jacks (i.e. No. 16-20 and 36-40) and ensure that the demanded pot. volts are greater or equal to the minimum safety values.

Line 0038 — IPD = Demanded jack pot. volts adjusted to overshoot the wall position, towards the model, by an increment IP.

Line 0039 — IMV = Required jack movement in pot. volts. (+ve towards the model and -ve away from the model).

Line 0041 — If the wall is being adjusted from its overshoot position (i.e. IP = 0) only allow the wall to move away from the model (i.e. IMV < 0). Set IMV = 0 if the jack tries to move towards the model. This technique ensures the wall is rigid to air pressure loads by eliminating jack mechanism backlash in the correct sense.

Line 0043 — Change the sign of all bottom wall movement demands for hardware compatability.

Lines 0045 - 0049 — Change the sign of top wall movement demands for all odd numbered jacks to accommodate the hardware configuration for the top wall control.

Line 0051 — Jump to line 0063 if this is the first pass (i.e. IF = 1).

Line 0053 — For jacks 1 and 21 do not check for jack jamming due to the small movement demands on these jacks positioned close to the wall anchor point (See figure 7).

Line 0055 — If the demanded jack movement (IML(L)) is more than 50 pot. volts and the jack potentiometer reading shows only a change of 10 or less pot. volts then warn the operator and abort the wall adjustment.

Line 0057 — If only one jack has yet to be correctly adjusted (IS = 39) increment ICNT for each pass through the adjustment cycle.

Line 0059	If IS = 39 and ICNT = 6 then warn the operator and abort the wall adjustment. One jack may have moved out of its potentiometer measuring range.
Line 0061	If the change in jack pot. volts between position samples is greater than 50 warn the operator and abort the wall adjustment. A jack may be moved the wrong way.
Line 0063	Load array IM with values of required jack movement.
Line 0064	If the required movement is less than or equal to ITOL go to statement 40 (line 0067).
Line 0067 - 0068	Increment IS by one and set IMV = 0.
Line 0069	Load jack (L) with direction information using Subroutine MOVE.
Line 0071	Set IF = 0 to indicate the first pass is complete.
Line 0072	If IS = 40 the wall is correctly adjusted.
Line 0074	Move the walls using Subroutine START.
Line 0075	If IS less than 40 repeat the wall adjustment cycle
Lines 0077 - 0080	If the wall has been correctly adjusted to the demanded contour (IP = 0) finish. If the wall has been correctly adjsuted to the overshoot contour (IP = 5) repeat the wall adjustment cycle with IP = 0.
Line 0081	Set IW = 0 for top wall condition.
Line 0082 - 0092	For all downstream jacks (Nos. 16-20 and 36-40) warn the operator when their pot. limit has been reached.

Subroutine INIT

Line 0002	NWJ = 40 = Number of wall jacks.
Line 0003	Set data operation as write before read IDI = 128.

Line 0004 - 0010	Generate command numbers ICOM for a motor stop directive to each jack and send them individually to the wind tunnel using Subroutine DIO.
Line 0007	Check the data has been correctly loaded.

Subroutine START

Line 0001	Subroutine statement with data transfer of the demanded jack pot. volts (PV) and the required movements or position errors (IM).
Line 0004 - 0006	Determine the average position error of the bottom wall in pot. volts and store the result in IAM.
Line 0007	RCT = number of clock counts between motor power on and off.
Line 0008	The upper limit of RCT is 600. Set by wall damage considerations.
Line 0010	The lower limit on RCT is 25. Set by resolution of the wall position measurements and speed considerations.
Line 0012 - 0014	If RCT > 50 load jacks 1 and 21 with a motor stop because they will overshoot by too large an amount.
Line 0016 - 0019	Generate a command number ICOM to start the motors (See Appendix A) and send it to the wind tunnel using Subroutine DIO.
Line 0020 - 0022	Variable time delay set by number of clock counts (RCT) at 100 Hz.
Line 0023	Stop all motors

Subroutine MOVE

Line 0001	Subroutine statement with data transfer of IJ (Jack number) and IMV (required jack movement).
Line 0002	Determine whether IMV is negative, positive or zero.

Lines 0003 - 0007	Load the jack direction data into IDR corresponding to either stop, forward-go or reverse-go.
Line 0008	All data operations are write before read (IDI = 128).
Line 0009	Generate a command number ICOM.
Line 0010	Send ICOM to jack IJ using Subroutine DIO.

Subroutine DIO

This subroutine performs the handshaking between computer and wind tunnel to transfer data on the digital I/O lines.

Line 0001	Subroutine statement with data transfer ICOM(Command number),IDI (Data operation) and INPUT (Data check)
Line 0002	Load Digital output buffer (address 167774) with command number 45 to ensure all motors are switched off.
Line 0003	Send command number ICOM to the wind tunnel.
Line 0004	Halt program until an external data accept signal is received on the digital I/O status register (handshaking between wind tunnel & computer).
Line 0006	Reset the Digital I/O status register to zero.
Line 0007	If IDl = 0 (Write only operation) finish.
Line 0009	Halt the program until an external data ready pulse is received on the digital I/O status register.
Line 0011	Store the contents of the Digital input register in INPUT.
Line 0012	Initialise the status register.

3.10 Program REAN

The main re-analysis program listed on Figure 4.10 reads test parameters and sequences subroutine calls. The program is very similar to TSWT, the main differences are:

1. The printout trigger is set to give an extended print-out (i.e. CL1 = 500).

2. All test parameters and run details are provided by the operator.

3. The program contains no loops and will run through only once per activation.

4. Subroutine WALL is not included.

5. Subroutine STAR is included to provide additional information during the re-analysis.

3.11 Program RUN

This program has to be activated before each session of TSWT runs, to ensure the data file RUN.DAT contains the current ambient conditions, run and file numbers.

The software is shown on Figure 4.11 and discussed below.

Line 0002	Define array TUN to hold run data.
Lines 0003 - 0004	Define File 2 as data file RUN.DAT.
Line 0005	Load array TUN with run data stored in RUN.DAT, record 1.
Lines 0007 - 0009	Indicate if only an increment of the record number is required (Yes: INC = 1).
Lines 0010 - 0011	Read thermocouple A-D channel and load TUN (3) with the current ambient temperature.
Line 0012	If INC = 1 jump to line 0025.
Line 0014 - 0021	Input ambient pressure (cm Hg), the run number and the file number as requested.
Lines 0022 - 0023	Load TUN (1) & TUN (2) with new values.
Line 0025	Increment the record number by one.
Line 0026	Write the modified contents of array TUN to RUN.DAT on disc.
Lines 0027 - 0028	Print the contents of array TUN as a check.

4. SYMBOLS

V	=	wall velocity perturbation ($V = U - U_\infty$).
U	=	local wall velocity.
U_∞	=	freestream velocity.
δ^*	=	boundary layer displacement thickness.
x	=	horizontal distance from the boundary layer origin (Inches).
R_x	=	Reynolds number based on boundary layer length (x).
Y	=	vertical distance from the tunnel centreline (positive upwards).
C_L	=	lift coefficient.
C_D	=	pressure drag coefficient.
C_m	=	pitching moment coefficient about the airfoil leading edge.
C_N	=	normal force coefficient.
C_c	=	chordwise force coefficient.
M	=	Freestream Mach number.
X	=	horizontal distance from leading edge (Inches).
Y	=	vertical distance from the chord line (Inches).

5. REFERENCES

1. M.J. Goodyer and S.W.D. Wolf — 'The Development of a Self-Streamlining Flexible Walled Transonic Test Section', AIAA Paper 80-0440, March 1980.

2. S.W.D. Wolf and M.J. Goodyer — 'Self-Streamlining Wind Tunnel-Low Speed Testing and Transonic Test Section Design', NASA CR-145257, October 1977.

3. M. Judd, S.W.D. Wolf and M.J. Goodyer — 'Analytical Work in Support of the Design and Operation of Two-Dimensional Self-Streamlining Test Sections', NASA CR-145019, July 1976.

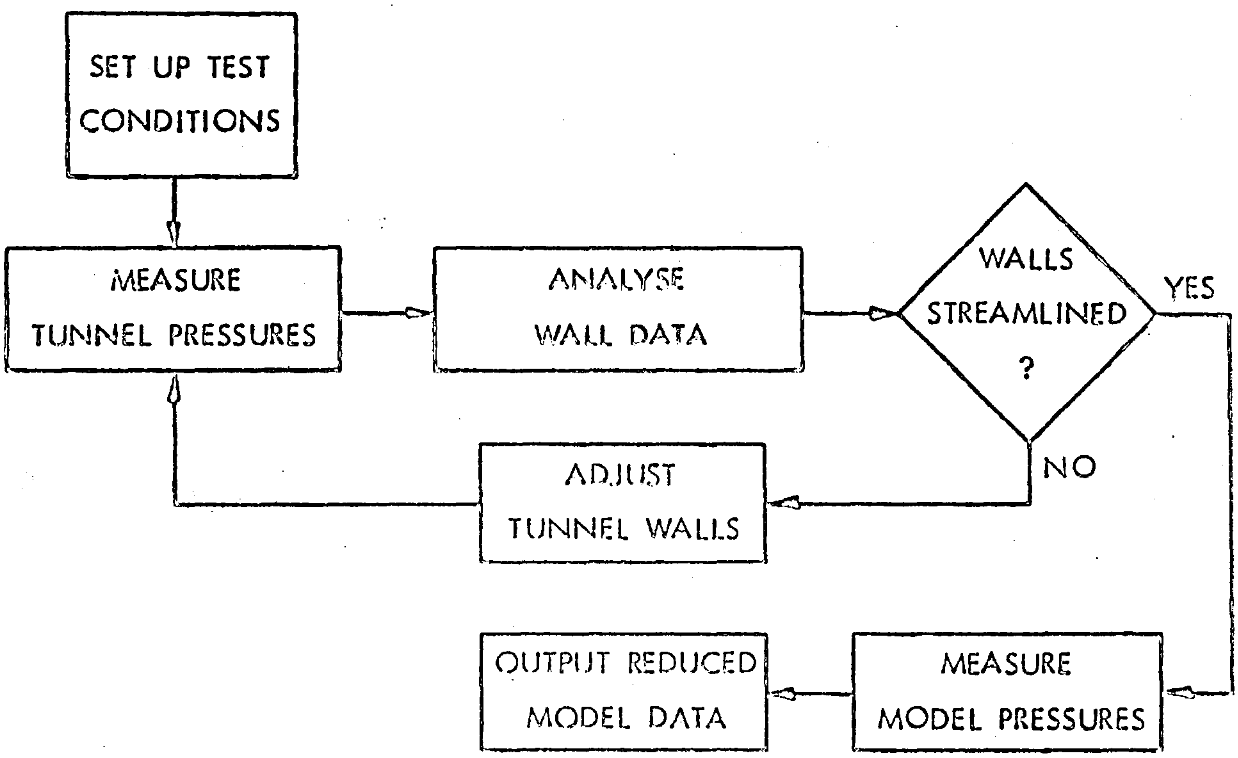


FIG. 1 SELF-STREAMLINING OPERATING PROCEDURE

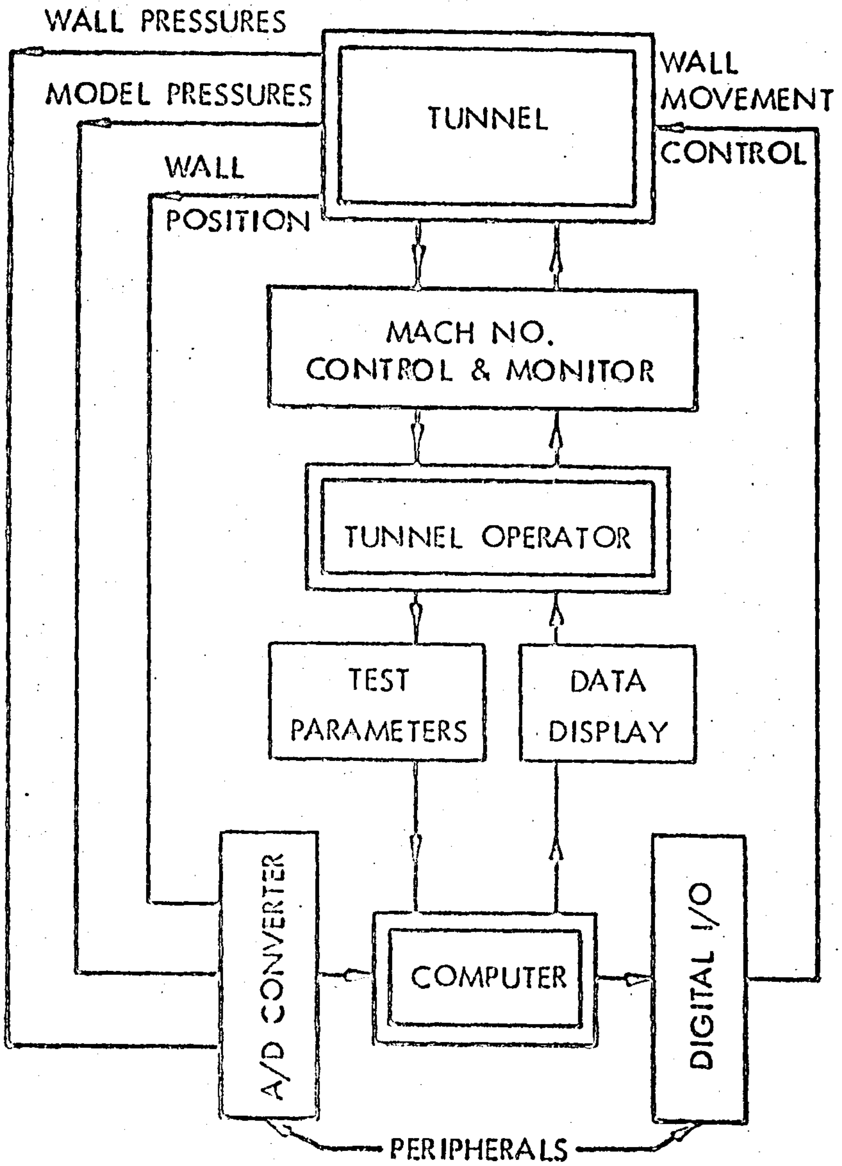
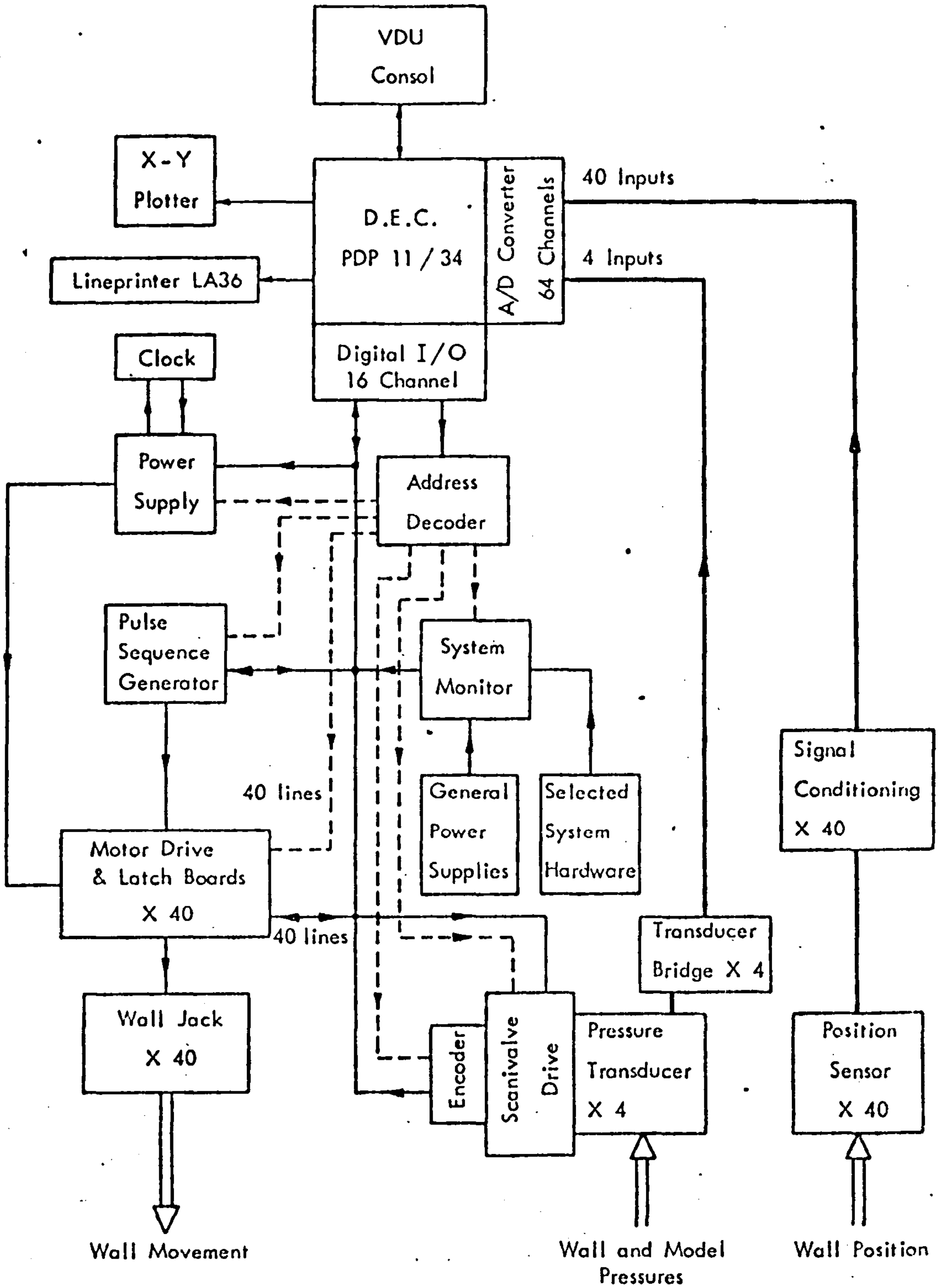


FIG. 2 TSWT CONTROL SYSTEM OUTLINE



Analogue signals

Digital signals { Information
Addressing

FIG. 3 T.S.W.T. CONTROL SYSTEM HARDWARE

Figure 4 Listings of TSWT Control Software

```
      SUBROUTINE WAS
      COMMON/ONE/NJ,MT,NR,CL1,B1,PR1,AK1,AK3,AN,R3,PP2,ITRN
      COMMON/TWO/RN(30),RS(30),W(30),X(30),P(30),Q(30),RD(20)
      COMMON/THREE/D(60),WTY(30),WBY(30),WL(30),B(50),PD(48,4)
      COMMON/FOUR/E(30),H(30),Y(30),G(30),WI(30),XI(30)
      COMMON/FIVE/U(30),V(30),DS(60),DET(60),CS(20)
      DIMENSION A(4),XB(4),C(30,4),Z(30),S(30),T(30)
      REAL N
      WRITE(7,5)
5     FORMAT(///5X,' WAS COMPUTING NOW !!')
      AK2 = AK1
      AK4 = AK3
      NJ1 = NJ-4
36    MM = 1
      M = NJ
      DO 100 I = 1,NJ
      IF(MM.EQ.NR) GO TO 10
      NT = (I-5)/6
      NT = NT * 6
      IF((I-5).EQ.NT) MM = MM +1
10    Q1 = (RD(MM) - RD(NR + 1))/PP2
      TEMP = -P(I)/Q1
      TEMP1 = B1*TEMP
      TEMP = SQRT(1-TEMP1)
      TEMP = TEMP-1
      U(I) = TEMP-X(I)
      E(I) = (AK3*U(I)/2)+X(I)
      TEP = -Q(I)/Q1
      TEP1 = B1*TEP
      TEP = SQRT(1-TEP1)
      TEP = TEP-1
      V(I) = W(I)-TEP
      H(I) = W(I)-(AK4*V(I)/2)
      WI(I) = (TEP+1)*(TEP+1)
      XI(I) = (TEMP+1)*(TEMP+1)
100   CONTINUE
      L = M-2
      DO 110 I = 1,L
110   Z(I) = (D(I)+D(I+1))/2
      DO 175 NN = 1,2
      NC = NN-1
65    I = 0
35    DO 120 J = 1,4
      KI = I+J
      A(J) = D(I+J)
15    IF (NC.EQ.0) GO TO 25
      XB(J) = V(I+J)
      GO TO 120
25    XB(J) = U(I+J)
120   CONTINUE
      V0 = (XB(3)-XB(2))/(A(3)-A(2))
      V1 = XB(2)-V0*A(2)
      V2 = 1/(A(4)-A(1))
      V3 = (XB(4)-V0*A(4)-V1)/((A(4)-A(2))*(A(3)-A(4)))
```

Figure 4.4.1

```
      CALL REDUCE
      CALL WAS
      CALL SUME
      IF(MT.EQ.0) GO TO 100
      CALL FORCE
100   CALL SET
      IF(PR1.EQ.1.) GO TO 200
      CALL WALL
200   IAM=1
      IF(IFN.EQ.50) WRITE(7,400)
400   FORMAT(//60('*')/'    REMEMBER TO COPY DATA FILES BEFORE NEXT RUN'
     C    /60('*'))
      IF(IFN.EQ.50) GO TO 250
      IFN=IFN+1
      IF(PR1.EQ.0.) GO TO 500
250   TUN(1)=ITRN+1
      TUN(2)=IFN
      IF(IFN.EQ.50) TUN(1)=ITRN
      IF(IFN.EQ.50) TUN(2)=4.
510   CALL ASSIGN(4,'RUN.DAT',0,'OLD',,)
      DEFINE FILE 4 (6,256,U,IJR)
      WRITE(4'1)(TUN(N),N=1,6)
      CALL CLOSE(4)
      END
TSWT
```

Figure 4.1.2

```
      SUBROUTINE DATA
C
C     AUTO SCAN
C
      DIMENSION DA(100),DB(100),DC(100),SO(4)
      DIMENSION IDATA(49,4)
      COMMON/ONE/NJ,MT,NR,CL1,B1,PR1,AK1,AK3,AN,R3,PP2,ITRN
      COMMON/TWO/RN(30),RS(30),W(30),X(30),P(40),Q(40),RD(20)
      COMMON/THREE/D(60),WTY(30),WBY(30),WL(30),B(50),PD(48,4)
      COMMON/FOUR/E(30),H(30),Y(30),G(30),WI(30),XI(30)
      COMMON/FIVE/U(30),V(30),DS(60),DET(60),CS(20)
      COMMON/SEVEN/PC(50)
      COMMON/EIGHT/TAB,AMP,IFN,IAM
      COMMON/NINE/IX(4)
      DATA IDATA(1,1),IDATA(1,2),IDATA(1,3),IDATA(1,4)/0,0,0,0/
C
C     DATA ACQUISITION
C
      CHD = 4.
      NJ1 = NJ-4
      M = NJ
      NR = 8
      IP1 = 2050
      SO(1) = .0331668
      SO(2) = .0142486
      SO(3) = .0134165
      SO(4) = .0141155
      IPO=0
37    IF(CL1.EQ.0) GO TO 36
      WRITE(5,31)
31    FORMAT(/'$  DATA INPUT FILE = ')
      CALL ASSIGN (4,,-1,'OLD',,)
      DEFINE FILE 4 (50,512,U,IJR)
      WRITE(7,32) IFN
32    FORMAT(40('*')/'      ITERATION RECORD NO. = ',I4/40('*'))
      READ(4'IFN)((IDATA(N,J),N=1,49),J=1,4)
      CALL CLOSE(4)
      GO TO 44
36    IDATA(1,1)=IX(1)
      IDATA(1,2)=IX(2)
      IDATA(1,3)=IX(3)
      IDATA(1,4)=IX(4)
      IF(CL1.EQ.0.AND.X(30).EQ.1)WRITE(5,800)
800   FORMAT(' SCANIVALVE GO'/13('='))
      DO 25 N=1,49
      IF(N.GT.1) GO TO 40
      IF(X(30).EQ.1.) GO TO 25
      CALL GAD(N,IDATA)
      WRITE(5,15)
15    FORMAT(/'   TURN ON WIND')
      READ(5,35) ISTOP
35    FORMAT(I2)
      GO TO 25
50    FORMAT(I3)
```

Figure 4.2.1.

```
40    CALL STEP
      IENC=CVSWG(IADC(20,1),1)
      IP2=IABS(IENC-IP1)
      IP1 = IENC
      IF(IP2.GT.200) GO TO 30
      WRITE(7,43)
43    FORMAT(/'   STEP ERROR')
      GO TO 180
30    CALL GAD(N,IDATA)
25    CONTINUE
      IX(1)=IDATA(1,1)
      IX(2)=IDATA(1,2)
      IX(3)=IDATA(1,3)
      IX(4)=IDATA(1,4)
      CALL ASSIGN(4,'ADC.DAT',0,'OLD',,)
      DEFINE FILE 4(50,512,U,IJR)
      WRITE(7,42) IFN
42    FORMAT(/'  ITERATION RECORD NO. = ',I4)
      WRITE(4'IFN)((IDATA(N,J),N=1,49),J=1,4)
      CALL CLOSE (4)
C
C     DATA REDUCTION
C
44    DO 45 J = 1,4
      DO 5 N = 2,49
      IDATA(N,J)= IDATA(1,J)-IDATA(N,J)
      NN = N-1
      PD(NN,J)= SO(J)*IDATA(N,J)
5     CONTINUE
45    CONTINUE
      DO 110 N = 1,2
      P(N) = 0.0
      Q(N) = 0.0
110   CONTINUE
      DO 115 N = 22,24
      P(N) = PD(19,2)
      Q(N) = PD(19,4)
115   CONTINUE
      DO 120 N = 3,21
      IA = N-2
      P(N) = PD(IA,2)
      Q(N) = PD(IA,4)
120   CONTINUE
      IT=21
      DO 210 N=25,33
      IT=IT+1
      IF(N.EQ.28) IT=IT+1
      P(N)=PD(IT,2)
      Q(N)=PD(IT,4)
210   CONTINUE
      RD(NR+1) = PD(48,1)
      RD(NR+2) = AMP
      RD(NR+3) = TAB
      AD1 = 0.
```

Figure 4.2.2.

```
      R1 = RD(NR+2) - RD(NR+1)
      DO 55 J = 1,NR
      ITR = 1 + ((J-1)*6)
      RD(J) = PD(ITR,1)
      AD1 = AD1 + RD(J)
      R2 = RD(NR+2)-RD(J)
      R3 = 0.28571*ALOG(R1/R2)
      PP1 = 5.0 * (EXP(R3)-1)
      TFM = SQRT(PP1)
      IF(J.EQ.1) TFM1=TFM
      TR = RD(J)/2.54
      IF(J.EQ.1) GO TO 55
      IF(ABS(TFM-TFM1).GT.0.01) WRITE(7,56)
56    FORMAT(/'  MACH NO. ERROR'/20('*'))
55    CONTINUE
      H1 = PD(1,2) + RD(1)
      H2 = PD(43,2) + RD(6)
      H3 = PD(20,2) + RD(4)
      H4 = PD(1,4) + RD(1)
      H5 = PD(43,4) + RD(6)
      H6 = PD(20,4) + RD(4)
      H7 = PD(2,1)
      H1 = H1/2.54
      H2 = H2/2.54
      H3 = H3/2.54
      H4 = H4/2.54
      H5 = H5/2.54
      H6 = H6/2.54
      H7 = H7/2.54
      AS = AD1/NR
      TR = AS/2.54
      SP = RD(NR+1)/2.54
      B1=AS
180   RETURN
      END
DATA
```

Figure 4.2.3.

```
      SUBROUTINE GAD (N,IDATA)
      DIMENSION IDATA(49,4),IDT(500)
      NS=15
      IEND = 0
      CALL SETR(4,0,50.,IEND)
      CALL LWAIT(0,IEND)
      IEND = 0
      IRTS=0
      CALL RTS(IDT,60,,NS,16,4,,2,IRTS,IDUM)
      CALL SETR(4,9,1.,IEND)
      CALL LWAIT(0,IRTS)
      DO 5 J = 1,4
      SUM = 0
      DO 70 NN = 1,NS
      IS = ((NN-1)*4)+J
      IDT(IS) = CVSWG(IDT(IS),1)
      SUM = SUM + IDT(IS)
70    CONTINUE
      IDATA(N,J) = SUM/NS
5     CONTINUE
      CALL SETR(-1,,,)
      RETURN
      END
GAD
```

```
      SUBROUTINE STEP
      CALL IPOKE("167774,16384)
      IEND=0
      CALL SETR(4,0,20.,IEND)
      CALL LWAIT(0,IEND)
      CALL IPOKE("167774,0)
      RETURN
      END
STEP
```

Figure 4.2.4.

```
      SUBROUTINE REDUCE
      DIMENSION DA(100),DB(100),DC(100),SO(4)
      DIMENSION IDATA(49,4)
      COMMON/ONE/NJ,MT,NR,CL1,B1,PR1,AK1,AK3,AN,R3,PP2,ITRN
      COMMON/TWO/RN(30),RS(30),W(30),X(30),P(40),Q(40),RD(20)
      COMMON/THREE/D(60),WTY(30),WBY(30),WL(30),B(50),PD(48,4)
      COMMON/FOUR/E(30),H(30),Y(30),G(30),WI(30),XI(30)
      COMMON/FIVE/U(30),V(30),DS(60),DET(60),CS(20)
      COMMON/SEVEN/PC(50)
      COMMON/EIGHT/TAB,AMP,IFN,IAM
C
C     DATA ACQUISITION
C
      AS=B1
      CHD = 4.
      NJ1 = NJ-4
      M = NJ
      NR = 8
      MM = 1
      R1 = RD(NR+2)-RD(NR+1)
      R2 = RD(NR+2) - AS
      R3 = 0.28571*ALOG(R1/R2)
      PP1 = 5.0*(EXP(R3)-1)
      AM1 = SQRT(PP1)
      B1 =SQRT(1-PP1)
      WRITE(5,90) AM1
      WRITE(7,90) AM1
 90   FORMAT(//'    MACH NO.  =  ',F8.4)
      PP2 = 1.0 + (0.25*PP1) + (0.025*PP1*PP1)
      CC1 = 1 + (0.2*PP1)
      CC2 = 2.5 * ALOG(CC1)
      CC3 = EXP(CC2)
      D1 = RD(NR+2)*8.998E-3/(273.15+RD(NR+3))
      D1 = D1/CC3
      AT = RD(NR+3)+273.15
      AT = AT/CC1
      CC4 = 0.92*ALOG(AT/273.15)
      V1 = 11.52*EXP(CC4)
      PP3 = (AS-RD(NR+1))/PP2
      U0 = SQRT(56.369*PP3/D1)
      R2 = U0 * D1 * 3218E4/V1
      R3 = R2 * CHD/12.
      WRITE(7,95) R3
 95   FORMAT(//'  REYNOLDS NO. -  ',E12.6)
      DO 60 J = 2,4
      MM=1
      DO 65 N = 1,48
      IF(MM.EQ.NR) GO TO 170
      NT = (N-5)/6
      NT = NT * 6
      IF((N-5).EQ.NT) MM = MM + 1
 170  Q1 = (RD(MM)-RD(NR+1))/PP2
      PD(N,J) = -PD(N,J)/Q1
 65   CONTINUE
```

Figure 4.3.1.

```
60    CONTINUE
      DO 500 J=16,24
      IJ=J+8
      JB=J+29
      IJB=J+22
      B(IJ)=PD(J,3)
      IF(J.GT.21) GO TO 500
      B(JB)=PD(IJB,3)
500   CONTINUE
      MM=1
      NP=1
      DO 600 K=2,42
      NT=(K-5)/6
      NT=NT*6
      NJJ=(K-7)/6
      NJJ=NJJ*6
      IF((K-5).EQ.NT) MM=MM+1
      IF((K-7).EQ.NJJ) GO TO 600
      Q1=(RD(MM)-RD(NR+1))/PP2
      B(NP)=(RD(MM)-PD(K,1))/Q1
      NP=NP+1
      IF(K.EQ.28) NP=33
600   CONTINUE
      CALL ASSIGN(2,'TUN.DAT',0,'OLD',,)
      DEFINE FILE 2 (12,256,U,IJR)
      IF(CL1.EQ.0.) IWF=1
      IF(CL1.EQ.0.) GO TO 77
      WRITE(7,75)
75    FORMAT(/'  FILE NO. ?'/'  1 FOR M=<.725'/'  2 FOR .725<M>.825')
      WRITE(7,76)
76    FORMAT('$ 3 FOR M=>.825    ANS = ')
      READ(7,35) IWF
35    FORMAT(I2)
77    READ(2'IWF) (DA(J),J=1,80)
      READ(2'11) (DB(J),J=1,33)
      NJ2 = NJ1 + NJ1
      READ(2'12)(PC(J),J=1,NJ2)
      CALL ASSIGN(3,'PAD.DAT',0,'OLD',,)
      DEFINE FILE 3 (50,512,U,INR)
      IF(IAM.EQ.1) GO TO 20
      WRITE(5,140)
140   FORMAT(//'$            WALL  RECORD? = ')
      READ(5,35) IR
      IF(CL1.EQ.0.) GO TO 41
      GO TO 40
20    IR=IFN-1
41    WRITE(7,30) IR
30    FORMAT(/'    WALL CONTOURS RECORD =',I4)
40    READ(3'IR) (DC(J),J=1,88)
      DO 150 K = 1,80
      IF(K.LE.20) WTY(K) = DA(K)
      IF(K.LE.40.AND.K.GT.20) WBY(K-20) = DA(K)
      IF(K.GT.40) DS(K-40) = DA(K)
150   CONTINUE
```

Figure 4.3.2.

```
      DO 155 J = 1,33
      IF(J.LE.9) CS(J) = DB(J)
      IF(J.GT.9) D(J-9) = DB(J)
155   CONTINUE
      READ(2'10) AK1,AK3
      DO 156 J = 1,NJ1
      NJJ = J + 2
      WL(J) = (D(NJJ+1)-D(NJJ-1))/2
156   CONTINUE
      DO 160 K = 1,88
      IF(K.LE.20) RS(K) = DC(K)
      IF(K.GT.20.AND.K.LE.40) RN(K-20) = DC(K)
      IF(K.GT.40.AND.K.LE.64) X(K-40) = DC(K)
      IF(K.GT.64) W(K-64) = DC(K)
160   CONTINUE
      DO 175 J = 1,NJ1
      WTY(J)=WTY(J)-RS(J)
      WTY(J)=WTY(J)/PC(J)
      WBY(J)=RN(J)-WBY(J)
      WBY(J)=WBY(J)/PC(J+NJ1)
175   CONTINUE
      CALL CLOSE(3)
      CALL CLOSE(2)
      X(30)=1.
      RETURN
      END
REDUCE
```

Figure 4.3.3.

```
      SUBROUTINE WAS
      COMMON/ONE/NJ,MT,NR,CL1,B1,PR1,AK1,AK3,AN,R3,PP2,ITRN
      COMMON/TWO/RN(30),RS(30),W(30),X(30),P(30),Q(30),RD(20)
      COMMON/THREE/D(60),WTY(30),WBY(30),WL(30),B(50),PD(48,4)
      COMMON/FOUR/E(30),H(30),Y(30),G(30),WI(30),XI(30)
      COMMON/FIVE/U(30),V(30),DS(60),DET(60),CS(20)
      DIMENSION A(4),XB(4),C(30,4),Z(30),S(30),T(30)
      REAL N
      WRITE(7,5)
5     FORMAT(///5X,' WAS COMPUTING NOW !!')
      AK2 = AK1
      AK4 = AK3
      NJ1 = NJ-4
36    MM = 1
      M = NJ
      DO 100 I = 1,NJ
      IF(MM.EQ.NR) GO TO 10
      NT = (I-5)/6
      NT = NT * 6
      IF((I-5).EQ.NT) MM = MM +1
10    Q1 = (RD(MM) - RD(NR + 1))/PP2
      TEMP = -P(I)/Q1
      TEMP1 = B1*TEMP
      TEMP = SQRT(1-TEMP1)
      TEMP = TEMP-1
      U(I) = TEMP-X(I)
      E(I) = (AK3*U(I)/2)+X(I)
      TEP = -Q(I)/Q1
      TEP1 = B1*TEP
      TEP = SQRT(1-TEP1)
      TEP = TEP-1
      V(I) = W(I)-TEP
      H(I) = W(I)-(AK4*V(I)/2)
      WI(I) = (TEP+1)*(TEP+1)
      XI(I) = (TEMP+1)*(TEMP+1)
100   CONTINUE
      L = M-2
      DO 110 I = 1,L
110   Z(I) = (D(I)+D(I+1))/2
      DO 175 NN = 1,2
      NC = NN-1
65    I = 0
35    DO 120 J = 1,4
      KI = I+J
      A(J) = D(I+J)
15    IF (NC.EQ.0) GO TO 25
      XB(J) = V(I+J)
      GO TO 120
25    XB(J) = U(I+J)
120   CONTINUE
      V0 = (XB(3)-XB(2))/(A(3)-A(2))
      V1 = XB(2)-V0*A(2)
      V2 = 1/(A(4)-A(1))
      V3 = (XB(4)-V0*A(4)-V1)/((A(4)-A(2))*(A(3)-A(4)))
```

Figure 4.4.1.

```
      V4 = (XB(1)-V0*A(1)-V1)/((A(1)-A(2))*(A(3)-A(1)))
      V6 = V2*(V3-V4)
      V5 = V4-V6*A(1)
      I = I+1
      P1 = A(2) + A(3)
      C(I,1) = V1-A(2)*A(3)*V5
      C(I,2) = V0+V5*P1-V6*A(2)*A(3)
      C(I,3) = V6*P1-V5
      C(I,4) = -V6
      IF (I.LT.(M-3)) GO TO 35
      LI = M-2
      DO 130 J = 2,LI
      Z0 = Z(J)
      Z02 = Z0*Z0
      Z03 = Z02*Z0
      SS = 0
      K = M- 3
      DO 140 I = 1,K
      Y1 =D(I+1)
      C0 = C(I,1)
      C1 = C(I,2)
      C2 = C(I,3)
      C3 = C(I,4)
      Y2 = D(I+2)
      Y2SQ = Y2 * Y2
      Y1SQ = Y1 * Y1
      S0 = C0+C1*Z0+C2*(Z02)+C3*(Z03)
      TEMP = ABS(Y2-Z0)/ABS(Y1-Z0)
      S1 = ALOG(TEMP)
      S2 = (C1+C2*Z0+C3*Z02)*(Y2-Y1)
      S3 = (C2+C3*Z0)*((Y2SQ)-(Y1SQ))/2
      S4 = C3*((Y2SQ*Y2)-(Y1SQ*Y1))/3
      SS = SS+S0*S1+S2+S3+S4
140   CONTINUE
      IF (NC.EQ.1) GO TO 45
      S(J) = SS/6.28319
      GO TO 130
45    T(J) = SS/6.28319
130   CONTINUE
175   CONTINUE
75    R = 0
      TT = 0
      S(1) = 0.0
      T(1) = 0.0
      DO 150 I =1,NJ1
      T0 = S(I)
      R0 = T(I)
      T1 = Z(I)
      I1 = I+1
      T2 = D(I1)
      T3 = S(I1)
      R3 = T(I1)
      T4 = Z(I1)
      I2 = I+2
```

Figure 4.4.2.

```
      T5 = D(I2)
      T6 = S(I2)
      R6 = T(I2)
      T7 = Z(I2)
      FS1 = (T6-T3)/(T7-T4)
      FS2 = (R6-R3)/(T7-T4)
      T2SQ = T2*T2
      T5SQ = T5*T5
      T8 = (FS1-(T0-T3)/(T1-T4))/(T7-T1)
      T9 = FS1 - T8 * T7
      P2 = (T3-T9*T4)*(T5-T2)+T8*((T5SQ*T5)-(T2SQ*T2))/3
      TT = TT+P2+(T9-T8*T4)*((T5SQ)-(T2SQ))/2
      R8 = (FS2-(R0-R3)/(T1-T4))/(T7-T1)
      R9 = FS2 - R8 *T7
      P3 = (R3-R9*T4)*(T5-T2)+R8*((T5SQ*T5)-(T2SQ*T2))/3
      F = E(I2)
      R = R+P3+(R9-R8*T4)*((T5SQ)-(T2SQ))/2
      Y(I2) = (AK3*TT)+(AK2*AK4*R)
      G(I2) = (AK4*R)+(AK1*AK3*TT)
      E(I2) = E(I2) + ((H(I2)-W(I2)+V(I2))*AK2)
      H(I2) = H(I2)+((F-U(I2)-X(I2))*AK1)
      Y(I2) = B1 * Y(I2)
      G(I2) = B1 * G(I2)
150   CONTINUE
      RETURN
      END
WAS
```

Figure 4.4.3.

```
      SUBROUTINE STAR
      DIMENSION DELTA(60)
      COMMON/ONE/NJ,MT,NR,CL1,B1,PR1,AK1,AK3,AN,R3,PP2,ITRN
      COMMON/TWO/RN(30),RS(30),W(30),X(30),P(30),Q(30),RD(30)
      COMMON/THREE/D(60),WTY(30),WBY(30),WL(30),B(50)
      COMMON/FOUR/E(30),H(30),Y(30),G(30),WI(30),XI(30)
      COMMON/FIVE/U(30),V(30),DS(60),DET(60),CS(20)
      PP1 = 1 - (B1*B1)
      CC1 = 1 + (.2*PP1)
      CC2 = 2.5 * ALOG(CC1)
      CC3 = EXP(CC2)
      D1 = RD(NR+2)*8.998E-3/(273.15+RD(NR+3))
      D1 = D1/CC3
      AT = RD(NR+3)+273.15
      AT = AT/CC1
      CC4=.92*ALOG(AT/273.15)
      V1 = 11.52*EXP(CC4)
      IF(CL1.EQ.0.) GO TO 56
      WRITE(7,5)
5     FORMAT(///5X,' DELTA STAR CALCS.'///'   TOP WALL'/)
      WRITE (7,10)
10    FORMAT(2X,' TAP NO.',2X,' DU/DX',5X,' MACH NO.',6X,' D*',9X' DD*')
56    NJ2 = NJ * 2
      RR1 = RD(NR+2) - RD(NR+1)
      MM = 1
      DO 2 N = 2,NJ2
      NN = N
      IF(N.GT.NJ) NN = N-NJ
      IF(NN.GT.(NJ-2)) GO TO 2
      IF(CL1.EQ.500..AND.N.EQ.(NJ+1)) WRITE (7,11)
11    FORMAT(//'   BOTTOM WALL'/)
      IF(CL1.EQ.500..AND.N.EQ.(NJ+1)) WRITE (7,10)
      IF(N.EQ.(NJ+1))  MM = 1
      IF(N.EQ.(NJ+1)) GO TO 2
      IF (N.EQ.NJ.OR.N.EQ.NJ2) GO TO 12
      X1 = D(NN) - D(NN-1)
      X2 = D(NN+1) - D(NN-1)
15    IF(MM.EQ.NR) GO TO 120
      NTS = N-5
      NT = NTS/6
      NT = NT*6
      IF(NT.EQ.NTS) MM=MM+1
120   IF(N.GT.NJ) GO TO 110
      SP1 = P(N)
      SP2 = P(N+1)
      SP3 = P(N-1)
      GO TO 100
110   NN = N-NJ
      SP1 = Q(NN)
      SP2 = Q(NN+1)
      SP3 = Q(NN-1)
100   PP3 = (SP1+RD(MM)-RD(NR+1))/PP2
      PP4 = (SP2+RD(MM)-RD(NR+1))/PP2
      PP5 = (SP3+RD(MM)-RD(NR+1))/PP2
```

Figure 4.5.1.

```
      U1 = SQRT(56.369*PP3/D1)
      U2 = SQRT(56.369*PP4/D1)
      U0 = SQRT(56.369*PP5/D1)
C
C     LOCAL MACH NO. CALCS
C
      R2 = RD(NR+2) - (SP1+RD(MM))
      R3 = 0.28571*ALOG(RR1/R2)
      PP1 = 5.0*(EXP(R3)-1)
      AM1 = SQRT(PP1)
      IF(N.GT.NJ) Q(NN-1)=AM1
      IF(N.GT.NJ) GO TO 23
      P(NN-1)=AM1
23    IF (N.EQ.2.OR.N.EQ.(NJ+2)) GO TO 22
      GO TO 33
22    R2 = U0*D1*32.18E6/V1
      R1 = (R2*X1)/12
      C1 = 0.142857*ALOG(R1)
      C2 = EXP(C1)
      C3 = 0.00127 * X1
      P6 = C3/C2
      GO TO 133
33    Y1 = U1 - U0
      Y2 = U2 - U0
      A1 = (Y2-((X2*Y1)/X1))/((X2*X2)-(X1*X2))
      B2 = (Y1-(A1*(X1*X1)))/X1
      D2 = (2*A1*X1) + B2
12    DO 3 M1 = 1,1000
      P1 = (25 + M1*100)/1E6
      NODE = 1
122   P2 = (U1*P1*D1*32.18E6)/V1
      C4 = 0.25 * ALOG(P2)
      C5 = EXP(C4)
      P3 = 0.0128/C5
      IF (N.EQ.NJ.OR.N.EQ.NJ2) GO TO 66
      S1 = P3-(3.4*P1*D2*12/U1)
      IF (N.EQ.3.OR.N.EQ.(NJ+3)) GO TO 77
      GO TO 88
66    S1 = P3-(3.4*P1*D3*12/U1)
      P4 = P5+(0.5*(S1+S2)*X1/12)
      GO TO 99
77    P4 = (S1*X1/12) + P6
      GO TO 99
88    P4 = P5+(0.5*(S1+S2)*X1/12)
99    IF (NODE.EQ.2) GO TO 155
      IF (P4.LT.P1) GO TO 111
3     CONTINUE
155   IF (P4.GT.P1) GO TO 144
111   P1 = P1 - 25E-6
      NODE = 2
      GO TO 122
144   P1 = P1 + 25E-6
      DELTA(N) = 16.8 *P1
      N2 = N-2
```

Figure 4.5.2.

```
0125          IF(N.GT.NJ) N2 = N-6
0127          DET(N2) = DELTA(N) - DS(N2)
0128          NNJ = NN-2
0129          IF(CL1.EQ.0.) GO TO 36
0131          WRITE(5,25) NNJ,D2,AM1,DELTA(N),DET(N2)
0132  25      FORMAT(5X,I2,4F12.4)
0133          IF(N.EQ.(NJ2-2)) WRITE(5,35) REY,FPD
0135  35      FORMAT(/' UNIT REYNOLDS NO. = ',F8.1,' D* FPG = ',F8.4/)
0136  36      P5 = P4
0137          S2 = S1
0138          IF (J-(NJ-1)) 2,44,55
0139  55      IF (J-(NJ2-1)) 2,44,2
0140  44      D3 = D2
0141          GO TO 2
0142  133     DELTA(N) = P6 * 16.8
0143          REY = R2/12
0144          FPD = DELTA(N)
0145  2       CONTINUE
      C
      C  WALL MACH NOS BETWEEN JACKS
      C
0146          IWALL=0
0147  350     DO 300 J=25,33
0148          SP1=P(J)
0149          IF(IWALL.NE.0) SP1=Q(J)
0151          R2=RD(NR+2)-(SP1+RD(5))
0152          R3=0.28571*ALOG(RR1/R2)
0153          PP1=5.0*(EXP(R3)-1)
0154          IF(PP1.LE.0) GO TO 300
0156          AM1=SQRT(PP1)
0157          IF(IWALL.EQ.0) P(J)=AM1
0159          IF(IWALL.NE.0) Q(J)=AM1
0161  300     CONTINUE
0162          IWALL=IWALL+1
0163          IF(IWALL.EQ.1) GO TO 350
0165          RETURN
0166          END
STAR
```

Figure 4.5.3.

```
      SUBROUTINE SUME
      COMMON/ONE/NJ,MT,NR,CL1,B1,PR1,AK1,AK3,AN,R3,PP2,ITRN
      COMMON/TWO/RN(30),RS(30),W(30),X(30),P(40),Q(40),RD(20)
      COMMON/THREE/D(60),WTY(30),WBY(30),WL(30),B(50),PD(48,4)
      COMMON/FOUR/E(30),H(30),Y(30),G(30),WI(30),XI(30)
      COMMON/FIVE/U(30),V(30),DS(60),DET(60),CS(20)
      DIMENSION UT(37),VT(37)
      IWT=1
      TOPI = 6.283185
      CHD = 4.
      IF(IWT.EQ.1) CHD=6.0
      OR=24.56
      AN1 = AN*.01745
      AX = (CHD*COS(AN1))/8.0
      AH = (CHD*SIN(AN1))/8.0
      IPP = 7
      IF(IWT.EQ.1) IPP = 4.33
      OR = OR - ((IPP-3)*AX)
      OT = 0
      TOT = 0.0
      NJ1 = NJ-4
      Y4 = 0.0
      Y3 = 3.0
      MN = 0.0
      EE = 0.
      F = 0.
      IJ = NJ-2
      DO 160 I = 3,IJ
      V1 = W(I)+1
      V2 = X(I)+1
      W(I) = V1 * V1
      X(I) = V2 * V2
      EE = EE + ABS(XI(I)-X(I))
      F = F + ABS(WI(I)-W(I))
  160 CONTINUE
      CET = EE/NJ1
      CEB = F/NJ1
      IF(CL1.EQ.0.)WRITE(5,170) CET,CEB
      WRITE(7,170) CET,CEB
  170 FORMAT(/10X,'    WALL CP ERROR'/5X,' TOP   -   ',F8.4,
     C    5X,' BOTTOM   -   ',F8.4)
      IF(CL1.EQ.0.) GO TO 95
      WRITE(7,5)
    5 FORMAT(10X,' RESIDUAL ERRORS')
      WRITE(7,10)
   10 FORMAT(7X,' X',8X,' U/UFS',6X,' V/UFS')
      DO 161 I = 1,37
      X1 = OR + ((I-19)*1.0)
   56 SUMU = 0.0
      SUMV = 0.0
      DO 162 J = 1,NJ1
      X2 = X1 - (D(J+2))
      Y1 = Y4+Y3-WTY(J)-DET(J)
      Y2 = Y4-Y3-WBY(J)+DET(J+NJ1)
```

Figure 4.6.1.

```
      P1 = X2 * X2
      P2 = P1 + (Y1*Y1)
      P3 = P1 + (Y2*Y2)
      R1 = X2/P2
      R2 = Y1/P2
      R3 = Y2/P3
      U1 = (U(J+2)*R2)+(V(J+2)*R3)
      U1 = (U1*WL(J))/TOPI
      V1 = (U(J+2)+V(J+2))*R1
      V1 = (V1*WL(J))/TOPI
      SUMU = SUMU + U1
      SUMV = SUMV + V1
162   CONTINUE
      IF (MN.GT.0) GO TO 98
      UT(I) = SUMU
      VT(I) = SUMV
      X3 = (I-19) * 1.0
      IF (OT.GT.0) GO TO 161
      WRITE(7,15) X3,UT(I),VT(I)
15    FORMAT(3F12.4)
161   CONTINUE
      WRITE (7,20)
20    FORMAT(/5X,' MODEL ERRORS',25X,' CP'/)
95    SO = 0.0
      SE = 0.0
      SM = 0.0
      DO 164 JJ = 1,9
      X1 = OR - ((3-JJ)*AX)
      Y4 = -(IPP-JJ)*AH
      MN = 1
      GO TO 56
98    AA2 = ATAN(SUMV/(1+SUMU))
      CP1 = 1.0 - ((1+SUMU)*(1+SUMU))
      IF (JJ.EQ.3) CP = CP1
      TOT = TOT + CP1
      IF (OT.GT.0) GO TO 104
      X1 = X1 - OR
      IF(CL1.EQ.0.) GO TO 104
      WRITE(7,25) X1,SUMU,SUMV,CP1
25    FORMAT(4F12.4)
104   IF (JJ.EQ.1) A1 = AA2
      IF (JJ.EQ.9) A2 = AA2
      SP = AA2 * (1-CS(JJ))
      IF (JJ.EQ.1.OR.JJ.EQ.9) GO TO 102
      MB = JJ/2
      MB = MB*2
      IF (MB.EQ.JJ) GO TO 101
      SO = SO + SP
      GO TO 164
101   SE = SE + SP
      GO TO 164
102   SM = SM + SP
164   CONTINUE
      CPE = TOT/9.0
```

Figure 4.6.2.

```
      P1 = SM+(2*SO)+(4*SE)
      P2 = (AX/3)*P1
      P3 = 2*P2
      A3 = (A1-A2) * 59.29578
      IF(CL1.NE.0) WRITE(7,35)
35    FORMAT(/10X,' EFFECT',25X,' DELTA CL')
      CL = TOPI * A1
      A1 = A1 * 59.29578
      IF(CL1.EQ.0) GO TO 58
      WRITE(7,30) A1,CL
30    FORMAT(/5X,' ALPHA ERROR = ',F8.4,' DEGREES',5X,F8.4)
      WRITE(7,40) A3,P3
40    FORMAT(/5X,' INDUCED CAMBER = ',F8.4,' DEGREES',2X,F8.4)
      WRITE(7,45) CP
45    FORMAT(/5X,' VEL.ERROR CP =',F8.4)
      WRITE (7,55) CPE,-CP
55    FORMAT(5X,' AVERAGE',6X,'=',F8.4,13X,F8.4///)
58    IF(CL1.EQ.0) WRITE(5,57) A1,A3,CPE
      IF(CL1.EQ.0) WRITE(7,57) A1,A3,CPE
57    FORMAT('  RESIDUALS = ',3F8.4)
      ISC=0
      IF(CET.LE..01)ISC=ISC+1
      IF(CEB.LE..01)ISC=ISC+1
      IF(ABS(A1).LE..015)ISC=ISC+1
      IF(ABS(A3).LE..07)ISC=ISC+1
      IF(ABS(CPE).LE..007)ISC=ISC+1
      IF(ISC.EQ.5)PR1=1.
      IF(CL1.EQ.0.AND.PR1.EQ.1.) WRITE(5,800)
      IF(PR1.EQ.1.) WRITE(7,810)
800   FORMAT(/6('*'),'TURN OFF WIND',6('*'))
810   FORMAT(//9X,23('$')/9X,'$  WALLS STREAMLINED  #'/9X,23('#'))
      RETURN
      END
SUME
```

Figure 4.6.3.

```
      SUBROUTINE FORCE
      COMMON/ONE/NJ,MT,NR,CL1,B1,PR1,AK1,AK3,AN,R3,PP2,ITRN
      COMMON/TWO/RN(30),RS(30),W(30),X(30),P(30),Q(30),RD(20)
      COMMON/THREE/D(60),WTY(30),WBY(30),WL(30),B(50),PD(48,4)
      COMMON/FOUR/E(30),H(30),Y(30),G(30),WI(30),XI(30)
      COMMON/FIVE/U(30),V(30),DS(60),DET(60),CS(20)
      DIMENSION  A(50),BB(50),AJ(50),AG(50),AH(50)
      REAL L5,L6,L7,J
      INTEGER S
      MT = MT+2
      CALL ASSIGN(2,'WING.DAT',0,'OLD',,)
      DEFINE FILE 2 (100,128,U,INR)
      READ(2'1) (AG(L),L=1,MT)
      READ(2'2) (AJ(L),L=1,MT)
      IHT = MT/2
      WRITE(7,30)
30    FORMAT(///15X,' NACA SECTION ANALYSIS'/20X,' 0012-64'/
     C    /'$                   RUN NO. = ')
      WRITE(7,7) ITRN
      WRITE(7,9)
9     FORMAT(/'$                        ALPHA = ')
      WRITE(7,12) AN
12    FORMAT('+',F6.2)
7     FORMAT('+',I5)
      DO 54 K = 2,(MT-1)
54    AH(K) = (AJ(K+1)-AJ(K-1))/2
      AH(1) = AJ(2)/2
      AH(IHT+1) = AJ(IHT+2)/2
      AH(IHT) = (.0012 - AJ(IHT-1))/2
      AH(MT) = (.0012 - AJ(MT-1))/2
      L5 =0
      DO 3 IQ = 1,MT
      IF (IQ.EQ.1) GO TO 540
      IF (IQ.EQ.IHT) GO TO 580
      IF (IQ.EQ.(IHT+1)) GO TO 630
      IF (IQ.EQ.MT) GO TO 530
      IF (IQ.GT.(IHT+1)) GO TO 780
      WF = (AG(IQ+1)-AG(IQ-1))/2
      L6 = WF*B(IQ)
      L5= L5-L6
      A(IQ) = -L6
      GO TO 3
540   WF = AG(IQ+1)/2
      L5 = -WF*B(IQ)
      A(IQ) = L5
      GO TO 3
580   WF = (1.0 - AG(IHT-1))/2
      L6=WF*B(IQ)
      L5=L5-L6
      A(IQ)=-L6
      CN = L5
      GO TO 3
630   WF = AG(IQ+1)/2
      L6 = WF*B(IQ)
```

Figure 4.7a.1.

```
      L5=L5+L6
      A(IQ) = L6
      GO TO 3
530   WF = (1.0 - AG(MT-1))/2
      L6 = WF*B(MT)
      L5 =L5+L6
      A(MT)= L6
      CNL = L5-CN
      GO TO 3
780   WF=(AG(IQ+1)-AG(IQ-1))/2
      L6 = WF*B(IQ)
      L5 = L5+L6
      A(IQ)= L6
3     CONTINUE
890   C3 = 0
      DO 40 IY = 1,MT
      IF (IY.GT.(IHT+1)) GO TO 980
      C5= (B(IY)*AH(IY))
      AH(IY)=C5
      C3 = C3+C5
      IF (IY.EQ.IHT) GO TO 970
      GO TO 40
970   CCP = C3
      GO TO 40
980   C5=(B(IY)*AH(IY))
      AH(IY) = C5
      C3 = C3 + C5
40    CONTINUE
      ST = C3-CCP
      IF(CL1.EQ.0.) GO TO 196
      WRITE (7,165)
165   FORMAT(////,10X,' UPPER SURFACE')
      WRITE (7,194)
194   FORMAT(2X,' %CHORD',3X,' CP LOCAL',3X,' CN LOCAL',3X,
     C    ' CC LOCAL',3X,' CM LOCAL')
196   C9=0.
      C8=0.
      DO 5 S=1,MT
      IF (S.EQ.(IHT+1)) GO TO 1240
      IF (S.GT.(IHT+1)) GO TO 1270
      BB(S) = (-A(S)*AG(S))+(AH(S)*AJ(S))
      C9 = C9+BB(S)
      TIT = AG(S)*100
      IF(CL1.EQ.0.) GO TO 5
      WRITE(7,185) TIT,B(S),A(S),AH(S),BB(S)
      GO TO 5
1240  IF(CL1.EQ.0.) GO TO 1270
      WRITE(7,195)
195   FORMAT(///10X,' LOWER SURFACE')
      WRITE (7,194)
1270  BB(S)=(A(S)*AG(S))+(AH(S)*AJ(S))
      C8= C8-BB(S)
      TIT = AG(S)*100
      IF(CL1.EQ.0.) GO TO 5
```

Figure 4.7a.2.

```
      WRITE (7,185) TIT,B(S),A(S),AH(S),-BB(S)
185   FORMAT(4X,F4.1,4F12.4)
5     CONTINUE
      ST1 = C8+C9
      WRITE(7,205)
205   FORMAT(//18X,'PRESSURE',2X,'SUCTION',2X,'TOTAL'/)
      WRITE(7,206)CN,CNL,L5
206   FORMAT(12X,'CN',2X,3F9.4)
      WRITE(7,207)CCP,ST,C3
207   FORMAT(12X,'CC',2X,3F9.4)
      WRITE(7,208)C9,C8,ST1
208   FORMAT(12X,'CM',2X,3F9.4)
      AT = AN * 0.017453
      CS1 = COS(AT)
      SN = SIN(AT)
      CL = (L5*CS1)-(C3*SN)
      CD = (C3*CS1) + (L5*SN)
      WRITE (7,225)
225   FORMAT(///22X,' WING PERFORMANCE'/18X,'CL',10X,'CD',10X,'CM')
      WRITE (7,235) CL,CD,ST1
235   FORMAT(10X,3F12.4)
      CALL CLOSE(2)
      END
FORCE
```

Figure 4.7a.3.

```
      SUBROUTINE FORCE
      COMMON/ONE/NJ,MT,NR,CL1,B1,FR1,AK1,AK3,AN,R3,PP2,ITRN
      COMMON/TWO/RN(30),RS(30),W(30),X(30),P(30),Q(30),RD(20)
      COMMON/THREE/D(60),WTY(30),WBY(30),WL(30),B(50),PD(48,4)
      COMMON/FOUR/E(30),H(30),Y(30),G(30),WI(30),XI(30)
      COMMON/FIVE/U(30),V(30),DS(60),DET(60),CS(20)
      DIMENSION  A(50),BB(50),AJ(50),AG(50),AH(50)
      REAL L5,L6,L7,J
      INTEGER S
      CALL ASSIGN(2,'NPL.DAT',0,'OLD',,)
      DEFINE FILE 2 (50,128,U,INR)
      READ(2'1) (AG(L),L=1,MT)
      READ(2'2) (AJ(L),L=1,MT)
      IHT = 32
      IF(CL1.EQ.0.) GO TO 31
      WRITE(7,30)
 30   FORMAT(///15X,' NPL SECTION ANALYSIS'/20X,'   9510'/
     C    /'$                     RUN NO. = ')
      WRITE(7,7) ITRN
      WRITE(7,9)
 9    FORMAT(/'$                     ALPHA = ')
      WRITE(7,12) AN
 12   FORMAT('+',F6.2)
 7    FORMAT('+',I5)
      B(31)=(B(30)+B(32))/2.
 31   DO 54 K = 2,(MT-1)
 54   AH(K) = (AJ(K+1)-AJ(K-1))/2
      AH(1) = AJ(2)/2
      AH(IHT+1) = AJ(IHT+2)/2
      AH(IHT) = (.0024)
      AH(MT) = (.0024)
      L5 =0
      DO 3 IQ = 1,MT
      IF (IQ.EQ.1) GO TO 540
      IF (IQ.EQ.IHT) GO TO 580
      IF (IQ.EQ.(IHT+1)) GO TO 630
      IF (IQ.EQ.MT) GO TO 530
      IF (IQ.GT.(IHT+1)) GO TO 780
      WF = (AG(IQ+1)-AG(IQ-1))/2
      L6 = WF*B(IQ)
      L5= L5-L6
      A(IQ) = -L6
      GO TO 3
 540  WF = AG(IQ+1)/2
      L5 = -WF*B(IQ)
      A(IQ) = L5
      GO TO 3
 580  WF = .5-(AG(IHT-1)/2)
      L6=WF*B(IQ)
      L5=L5-L6
      A(IQ)=-L6
      CN = L5
      GO TO 3
 630  WF = AG(IQ+1)/2
```

Figure 4.7b.1.

```
      L6 = WF*B(IQ)
      L5=L5+L6
      A(IQ) = L6
      GO TO 3
530   WF = .5-(AG(MT-1)/2)
      L6 = WF*B(MT)
      L5 =L5+L6
      A(MT)= L6
      CNL = L5-CN
      GO TO 3
780   WF=(AG(IQ+1)-AG(IQ-1))/2
      L6 = WF*B(IQ)
      L5 = L5+L6
      A(IQ)= L6
3     CONTINUE
890   C3 = 0
      DO 40 IY = 1,MT
      IF (IY.GT.(IHT+1)) GO TO 980
      C5= (B(IY)*AH(IY))
      AH(IY)=C5
      C3 = C3+C5
      IF (IY.EQ.IHT) GO TO 970
      GO TO 40
970   CCP = C3
      GO TO 40
980   C5=(B(IY)*AH(IY))
      AH(IY) = C5
      C3 = C3 + C5
40    CONTINUE
      ST = C3-CCP
      IF(CL1.EQ.0.) GO TO 196
      WRITE (7,165)
165   FORMAT(////,10X,' UPPER SURFACE')
      WRITE (7,194)
194   FORMAT(2X,' %CHORD',3X,' CP LOCAL',3X,' CN LOCAL',3X,
     C     ' CC LOCAL',3X,' CM LOCAL')
196   C9=0.
      C8=0.
      DO 5 S=1,MT
      IF (S.EQ.(IHT+1)) GO TO 1240
      IF (S.GT.(IHT+1)) GO TO 1270
      BB(S) = (-A(S)*AG(S))+(AH(S)*AJ(S))
      C9 = C9+BB(S)
      TIT = AG(S)*100
      IF(CL1.EQ.0.) GO TO 5
      WRITE(7,185) TIT,B(S),A(S),AH(S),BB(S)
      GO TO 5
1240  IF(CL1.EQ.0.) GO TO 1270
      WRITE(7,195)
195   FORMAT(///10X,' LOWER SURFACE')
      WRITE (7,194)
1270  BB(S)=(A(S)*AG(S))+(AH(S)*AJ(S))
      C8= C8-BB(S)
      TIT = AG(S)*100
```

Figure 4.7b.2.

```
      IF(CL1.EQ.0.) GO TO 5
      WRITE (7,185) TIT,B(S),A(S),AH(S),-BB(S)
185   FORMAT(4X,F4.1,4F12.4)
5     CONTINUE
      ST1 = C8+C9
      IF(CL1.EQ.0) GO TO 209
      WRITE(7,205)
205   FORMAT(//18X,'PRESSURE',2X,'SUCTION',2X,'TOTAL'/)
      WRITE(7,206)CN,CNL,L5
206   FORMAT(12X,'CN',2X,3F9.4)
      WRITE(7,207)CCP,ST,C3
207   FORMAT(12X,'CC',2X,3F9.4)
      WRITE(7,208)C9,C8,ST1
208   FORMAT(12X,'CM',2X,3F9.4)
209   AT = AN * 0.017453
      CS1 = COS(AT)
      SN = SIN(AT)
      CL = (L5*CS1)-(C3*SN)
      CD = (C3*CS1) + (L5*SN)
      WRITE (7,225)
225   FORMAT(22X,' WING PERFORMANCE'/18X,'CL',10X,'CD',10X,'CM')
      WRITE (7,235) CL,CD,ST1
235   FORMAT(10X,3F12.4)
      CALL CLOSE(2)
      END
FORCE
```

Figure 4.7b.3.

```
      SUBROUTINE SET
      DIMENSION DA(100)
      COMMON/ONE/NJ,MT,NR,CL1,B1,PR1,AK1,AK3,AN,R3,PP2,ITRN
      COMMON/TWO/RN(30),RS(30),W(30),X(30),P(40),Q(40),RD(20)
      COMMON/THREE/D(60),WTY(30),WBY(30),WL(30),B(50),PD(48,4)
      COMMON/FOUR/E(30),H(30),Y(30),G(30),WI(30),XI(30)
      COMMON/FIVE/U(30),V(30),DS(60),DET(60),CS(20)
      COMMON/SEVEN/PC(50)
      COMMON/EIGHT/TAB,AMP,IFN,IAM
      DATA Y(1),Y(2),G(1),G(2),Y(23),Y(24),G(23),G(24)/0.,0.,0.,
     C     0.,0.,0.,0.,0./
      CALL ASSIGN(3,'PAD.DAT',0,'OLD',,)
      DEFINE FILE 3 (50,512,U,INR)
      CALL ASSIGN(2,'NPL.DAT',0,'OLD',,)
      DEFINE FILE 2 (50,128,U,INR)
C
C       DATA OUTPUT
C
      NJ1 = NJ-4
      IWT=1
      IF(CL1.EQ.0.) GO TO 12
      WRITE(7,65)
 65   FORMAT(/'  TRANSDUCER  OUTPUT'//'  CP VALUES CHANNELS 2-4'/
     C     7X,' 1',7X,' 2',7X,' 3',7X,' 4')
      WRITE(7,70) (N,(PD(N,J),J=1,4),N=1,48)
 70   FORMAT(I3,4F9.4)
      WRITE(7,41) ITRN
 41   FORMAT(//'  RUN ',I4,'  OUTPUT'/12X,' EXT VEL.',7X,' MOVEMENT',
     C     8X,' Y CO-ORD')
      DO 45 J = 1,NJ
      IF(J.LT.3.OR.J.GT.22) GO TO 63
      WMT = -WTY(J-2)
      WMB = -WBY(J-2)
      GO TO 62
 63   WMT = 0.0
      WMB = 0.0
 62   WRITE(7,50) D(J),E(J),H(J),Y(J),G(J),WMT,WMB
 50   FORMAT(7F8.4)
 45   CONTINUE
      WRITE(7,300)ITRN
 300  FORMAT(//'    RUN',I4,' DATA'/15X,' TOP WALL')
      WRITE(7,320)
 320  FORMAT(/5X,' JACK',5X,' MACH NO.')
      TMN=0
      BMN=0
      CALL ASSIGN(6,'WALM.DAT',0,'OLD',,)
      DEFINE FILE 6 (20,256,U,INR)
      WRITE(7,21)
 21   FORMAT(//'$   MACH NO. RECORD (1-20) = ')
      READ(7,22)IMR
 22   FORMAT(I6)
      DO 23 J=2,21
      X(J)=P(J)
      X(J+21)=Q(J)
```

Figure 4.8.1.

```
23    CONTINUE
      WRITE(6'IMR)(X(J),J=1,42)
      CALL CLOSE(6)
      GMN=0
12    DO 170 J = 1,NJ1
        IS1 = RS(J) + ((Y(J+2))*PC(J))
      DA(J)=IS1
      IP = RS(J)
      YM=P(J+1)
      WM = -WTY(J)-DET(J)
      IF (DA(16).LT.123.) DA(16)=123.
      IF (DA(17).LT.178.) DA(17)=178.
      IF (DA(18).LT.161.) DA(18)=161.
      IF (DA(19).LT.140.) DA(19)=140.
      IF (DA(20).LT.216.) DA(20)=216.
      IF(CL1.EQ.0.)GO TO 170
      WRITE(7,840)J,WM,YM,Y(J+2),IP,IS1
840   FORMAT(7X,I2,11X,3F11.4,2(7X,I3))
      IF(J.LT.5.OR.J.GT.13)GO TO 170
      IBJ=J+20
      WRITE(7,810)P(IBJ)
810   FORMAT(13X,F11.4)
170   CONTINUE
      IF(CL1.GT.0.) WRITE(7,20)
20    FORMAT(///15X,' BOTTOM WALL')
      DO 180 J = 1,NJ1
      IS1 = RN(J) - ((G(J+2))*PC(J+NJ1))
      DA(J+NJ1)=IS1
      IP = RN(J)
      WM=-WBY(J)+DET(J+NJ1)
      NIJ=J+16
      YM=Q(J+1)
      IF (DA(36).LT.209.) DA(36)=209.
      IF (DA(37).LT.143.) DA(37)=143.
      IF (DA(38).LT.136.) DA(38)=136.
      IF (DA(39).LT.183.) DA(39)=183.
      IF (DA(40).LT.270.) DA(40)=270.
      IF(CL1.EQ.0.)GO TO 180
      WRITE(7,840)J,WM,YM,G(J+2),IP,IS1
      GO TO 180
      IF(J.GT.19) GO TO 180
      AMN=(P(J+1)-Q(J+1))/2
      GMN=GMN+(AMN*AMN)
      TMN=TMN+(P(J+1)*P(J+1))
      BMN=BMN+(Q(J+1)*Q(J+1))
      IF(J.LT.5.OR.J.GT.13) GO TO 180
      IBJ=J+20
      WRITE(7,810)Q(IBJ)
180   CONTINUE
      KT = 2*NJ1
      DO 190 J = 1,NJ
190   DA(J+KT) = E(J)
      KS = KT + NJ
      DO 200 J = 1,NJ
```

Figure 4.8.2.

```
200   DA(J+KS) = H(J)
      IF(CL1.EQ.0.)WRITE(5,30) IFN
      WRITE(7,30) IFN
30    FORMAT(/'     WALL & MODEL OUTPUT RECORD NO. = ',I4//50('=')//)
35    FORMAT(I2)
      ND = 2 * (NJ+NJ1)
      WRITE(3'IFN) (DA(J),J=1,ND)
      MT2 = MT + 2
      IF(IWT.EQ.1) MT2=MT
      WRITE(2'IFN) (B(J),J=1,MT2)
      CALL CLOSE(2)
      CALL CLOSE(3)
      RETURN
      END
SET
```

Figure 4.8.3.

```
      SUBROUTINE WALL
C
C     ONLINE WALL EXERCISER
C
      COMMON/EIGHT/TAB,AMP,IFN,IAM
      NWJ=40
      DIMENSION PC(50),IM(50),ADC(50),PV(50)
      DIMENSION PL(10)
      DATA PL/130.,178.,161.,140.,216.,209.,143.,136.,183.,270./
      CALL ASSIGN(3,'PAD.DAT',0,'OLD',,)
      DEFINE FILE 3 (50,512,U,INR)
      READ(3'IFN) (PV(J),J=1,NWJ)
      CALL CLOSE(3)
C
C     CALCULATE MOVEMENT
C
      IF=1
      IP=5
      CALL INIT
   50 IW=0
      ICNT=0
      IS=0
      ITOL=3
      DO 25 L=1,NWJ
      IAI=L+23
      IJP=CVSWG(IADC(IAI))
      IPV=IJP*.24414
      IMOVE=ADC(L)-IPV
      ADC(L)=IPV
      IF(IPV.GE.980) WRITE(5,200) L
  200 FORMAT(/' POT LIMIT REACHED ON JACK ',I4)
      IF(IPV.GE.980) STOP
      IF(L.LE.15) GO TO 60
      IF(L.GT.20.AND.L.LE.35) GO TO 60
      IF(L.EQ.36) IW=15
      NDJ=L-(15+IW)
      IF(PV(L).LT.PL(NDJ)) PV(L)=PL(NDJ)
   60 IPD=PV(L)-IP
      IMV=IPV-IPD
      IJ=L
      IF(IP.EQ.0.AND.IMV.GT.0)IMV=0
      IF(L.GT.20) IMV=-IMV
      IF(L.GT.20) GO TO 90
      IEV=IJ/2
      IEV=IEV+IEV
      IF(IEV.NE.IJ) IMV=-IMV
   90 IF(IF.EQ.1)GO TO 300
      IF(L.EQ.1.OR.L.EQ.21) GO TO 110
      IF(IABS(IM(L)).GT.50.AND.IABS(IMOVE).LE.10) GO TO 400
      IF(IS.GE.39)ICNT=ICNT+1
      IF(IS.GE.39.AND.ICNT.EQ.6) GO TO 500
  110 IF(IABS(IMV)-IABS(IM(L)).GT.50) GO TO 500
  300 IM(L)=IMV
      IF(IABS(IMV).LE.ITOL) GO TO 40
```

Figure 4.9.1.

```
0066          GO TO 35
0067   40     IS=IS+1
0068          IMV=0
0069   35     CALL MOVE(IJ,IMV)
0070   25     CONTINUE
0071          IF=0
0072          IF(IS.EQ.40) GO TO 70
0074          CALL START(PV,IM)
0075          IF(IS.LT.40) GO TO 50
0077   70     IF(IP.EQ.0) GO TO 100
0079          IP=0
0080          GO TO 50
0081   100    IW=0
0082          DO 85 J=16,40
0083          IF(J.GT.20.AND.J.LT.36) GO TO 85
0085          IF(J.EQ.36)IW=15
0087          ILIM=J-(15+IW)
0088          IE=PV(J)-PL(ILIM)
0089          IF(IABS(IE).LE.ITOL) WRITE(5,80) J
0091   80     FORMAT('  LOWER POT LIMIT REACHED ON JACK ',I4)
0092   85     CONTINUE
0093          GO TO 250
0094   400    WRITE(5,150) L
0095   150    FORMAT(//'  JACK ',I4,'  JAMMED!!!!!!!!!!!')
0096          GO TO 250
0097   500    WRITE(5,160) L
0098   160    FORMAT(//'  WALL OUT OF CONTROL AT JACK ',I4/40('*'))
0099   250    RETURN
0100          END
WALL
```

```
0001          SUBROUTINE INIT
       C
       C      INITIALISE TSWT CONTROL SYSTEM
       C
0002          NWJ=40
0003          IDI=128
0004          DO 10 J=1,NWJ
0005          ICOM=J+IDI+0
0006          CALL DIO(ICOM,IDI,INPUT)
0007          IF(INPUT.NE.0) WRITE(5,30) J
0009   30     FORMAT(/'  JACK ',I4,'  I/O ERROR')
0010   10     CONTINUE
0011          RETURN
0012          END
INIT
```

Figure 4.9.2.

```
0001          SUBROUTINE START(PV,IM)
0002          DIMENSION PV(50),IM(50)
0003          ISUM=0
0004          DO 10 J=21,40
0005    10    ISUM=ISUM+IABS(IM(J))
0006          IAM=ISUM/20
0007          RCT=(IAM/2)*25.
0008          IF(RCT.GT.600.) RCT=600.
0010          IF(RCT.LT.25.) RCT=25.
0012          IF(RCT.GT.50.) CALL DIO(129,128,INPUT)
0014          IF(RCT.GT.50.) CALL DIO(149,128,INPUT)
        C
        C     START ALL MOTORS
        C
0016          IPSA=45
0017          IDI=0
0018          ICOM=IPSA+IDI+4096
0019          CALL DIO(ICOM,IDI,INPUT)
        C
        C     WAIT FOR WALL TO MOVE
        C
0020          IEND=0
0021          CALL SETR(5,0,RCT,IEND)
0022          CALL LWAIT(0,IEND)
        C
        C     STOP ALL MOTORS
        C
0023          CALL DIO(45,IDI,INPUT)
0024          RETURN
0025          END
START
```

```
0001          SUBROUTINE MOVE(IJ,IMV)
0002          IF(IMV-0) 15,10,5
0003    5     IDR=3072
0004          GO TO 20
0005    10    IDR=0
0006          GO TO 20
0007    15    IDR=2560
0008    20    IDI=128
0009          ICOM=IJ+IDI+IDR
        C
        C     LOAD COMMAND
        C
0010          CALL DIO(ICOM,IDI,INPUT)
0011          RETURN
0012          END
MOVE
```

```
0001          SUBROUTINE DIO(ICOM,IDI,INPUT)
0002          CALL IPOKE("167774,45)
0003          CALL IPOKE("167774,ICOM)
0004    10    IF(IPEEK("167770).GE.0) GO TO 10
0006          CALL IPOKE("167770,0)
0007          IF(IDI.EQ.0) GO TO 20
0009    30    IF(IPEEK("167770).NE.128) GO TO 30
0011          INPUT=IPEEK("167772)
0012          CALL IPOKE("167770,0)
0013    20    RETURN
0014          END
DIO
```

Figure 4.9.3.

```
      PROGRAM REAN
C
C     MAIN CONTROL PROGRAM
C
      COMMON/ONE/NJ,MT,NR,CL1,B1,PR1,AK1,AK3,AN,R3,PP2,ITRN
      COMMON/TWO/RN(30),RS(30),W(30),X(30),P(40),Q(40),RD(20)
      COMMON/THREE/D(60),WTY(30),WBY(30),WL(30),B(50),PD(48,4)
      COMMON/FOUR/E(30),H(30),Y(30),G(30),WI(30),XI(30)
      COMMON/FIVE/U(30),V(30),DS(60),DET(60),CS(20)
      COMMON/SEVEN/PC(50)
      COMMON/EIGHT/TAB,AMP,IFN,IAM
      CL1 = 500.
      NR = 8
      NJ = 24
      WRITE(7,5)
 5    FORMAT(10X,' UNIVERSITY OF SOUTHAMPTON'/5X,
     C     ' TRANSONIC SELF-STREAMLINING WIND TUNNEL'/20X,8('*'))
      WRITE(7,10)
 10   FORMAT(/17X,' DATA REANALYSIS'/17X,15('*'))
      CALL IDATE(I,J,K)
      WRITE(7,15) J,I,K
 15   FORMAT(19X,I2,2('-',I2))
      WRITE(7,40)
 40   FORMAT(/'$             RUN NO.  =  ')
      READ(7,45) ITRN
 45   FORMAT(I4)
      WRITE(7,110)
 110  FORMAT(/'$             FILE NO. = ')
      READ(7,45) IFN
      WRITE(7,50)
 50   FORMAT(/'$           MODEL ALPHA (DEG)  =  ')
      READ(7,60) AN
 60   FORMAT(F9.4)
      WRITE(7,70)
 70   FORMAT(/'$       NO. OF MODEL TAPS ? ')
      READ(7,45) MT
      WRITE(7,80)
 80   FORMAT(/'$    INPUT 1 FOR AUTO IN-FILE SELECTION -')
      READ(7,45) IAM
      WRITE(7,85)
 85   FORMAT(/' INPUT AMBIENT CONDITIONS'//'$     TEMP (DEG.C) = ')
      READ(7,25) TAB
 25   FORMAT(F8.4)
      WRITE(7,90)
 90   FORMAT('$    PRES. (CM HG) = ')
      READ(7,25) AMP
      CALL DATA
      CALL REDUCE
      CALL WAS
      CALL STAR
      CALL SUME
      IF(MT.EQ.0) GO TO 100
      CALL FORCE
 100  CALL SET
      END
REAN
```

Figure 4.10.1.

```
      PROGRAM RUN
      DIMENSION TUN(5)
      CALL ASSIGN(2,'RUN.DAT',0,'OLD',,)
      DEFINE FILE 2 (6,256,U,IJR)
      READ(2'1)(TUN(J),J=1,5)
      WRITE(5,25)
25    FORMAT('$    INPUT 1 FOR RECORD NO. INCREMENT    ')
      READ(5,35)INC
35    FORMAT(I6)
      IAD=CVSWG(IADC(21))
      TUN(3)=(IAD-2045)/8.15
      IF(INC.EQ.1) GO TO 20
      WRITE(5,5)
5     FORMAT(//'  GOOD MORNING'//' INPUT PRES.')
      READ(5,45)TUN(4)
45    FORMAT(F8.4)
      WRITE(5,15)
15    FORMAT(/'  INPUT NEXT RUN NO. & FILE NO.')
      READ(5,10)IR,IF
10    FORMAT(2I6)
      TUN(1)=IR
      TUN(2)=IF
      GO TO 30
20    TUN(2)=TUN(2)+1
30    WRITE(2'1)(TUN(J),J=1,5)
      WRITE(5,55)(TUN(J),J=1,5)
55    FORMAT(/'  RUN DATA STORED '/5F8.2)
      CALL CLOSE(2)
      END
RUN
```

Figure 4.11.1.

Figure 5 A sample output of the control software OFLEX

UNIVERSITY OF SOUTHAMPTON
TRANSONIC SELF-STREAMLINING WIND TUNNEL
****

AUTO MODE
22- 1-82

RUN NO. = 407

MODEL ALPHA (DEG) = 4.0

AMBIENT CONDITIONS
TEMP = 22.25 PRES(CM HG) = 76.65

ITERATION RECORD NO. = 31

MACH NO. = 0.5945

REYNOLDS NO. - 0.118538E+07

RECORD = 30

WALL CP ERROR
TOP - 0.0061 BOTTOM - 0.0055
RESIDUALS = 0.0287 -0.0042 0.0014

WALL & MODEL OUTPUT RECORD NO. = 31

==

ITERATION RECORD NO. = 32

MACH NO. = 0.5948

REYNOLDS NO. - 0.118590E+07

RECORD = 31

WALL CP ERROR
TOP - 0.0026 BOTTOM - 0.0038
RESIDUALS = -0.0163 0.0051 0.0000

WALL & MODEL OUTPUT RECORD NO. = 32

==

Figure 5.1.

```
ITERATION RECORD NO. =   33

MACH NO.  =    0.5966

REYNOLDS NO. -  0.118846E+07

      RECORD =  32

              WALL CP ERROR
    TOP  -    0.0029      BOTTOM  -    0.0028
RESIDUALS =  -0.0066  0.0101 -0.0017

      #######################
      #  WALLS STREAMLINED  #
      #######################


WALL & MODEL OUTPUT RECORD NO. =   33

   ==================================================
```

Figure 5.2.

Figure 6. A sample output of the
re-analysis software QRLEX

```
RUN ORLEX
         UNIVERSITY OF SOUTHAMPTON
    TRANSONIC SELF-STREAMLINING WIND TUNNEL
                   ********

                DATA REANALYSIS
               ***************
                 11- 2-82

             RUN NO.  =  389

             FILE NO. = 21

           MODEL ALPHA (DEG)  =  3.0

       NO. OF MODEL TAPS ? 50

     INPUT 1 FOR AUTO IN-FILE SELECTION -1

INPUT AMBIENT CONDITIONS
     TEMP (DEG.C) = 22.0
    PRES. (CM HG) = 76.32

  DATA INPUT FILE = *ADC.DAT
***************************************
     ITERATION RECORD NO. =   21
***************************************



   MACH NO.  =     0.8038


 REYNOLDS NO. -  0.143046E+07

 FILE NO. ?
 1 FOR M=<.725
 2 FOR .725<M>.825
 3 FOR M=>.825    ANS = 2

   WALL CONTOURS RECORD =  20


     WAS COMPUTING NOW !!
```

Figure 6.1.

DELTA STAR CALCS.

TOP WALL

TAP NO.	DU/DX	MACH NO.	D*	DD*
1	0.3870	0.8238	0.0155	-0.0005
2	-3.0340	0.8057	0.0227	-0.0004
3	0.6563	0.7998	0.0290	-0.0000
4	0.1942	0.8092	0.0344	-0.0009
5	1.9132	0.8012	0.0395	-0.0012
6	14.1533	0.8127	0.0403	-0.0042
7	32.5273	0.8366	0.0386	-0.0076
8	54.5744	0.8975	0.0349	-0.0130
9	55.2920	0.9884	0.0302	-0.0189
10	-2.2115	1.0633	0.0294	-0.0218
11	-53.4752	0.9817	0.0344	-0.0189
12	-37.5962	0.9032	0.0424	-0.0130
13	-17.6038	0.8758	0.0483	-0.0088
14	-10.6314	0.8436	0.0571	-0.0038
15	-4.2687	0.8159	0.0672	0.0013
16	-0.3833	0.8144	0.0735	0.0029
17	0.1915	0.8130	0.0781	0.0033
18	0.5746	0.8159	0.0823	0.0029
19	0.1913	0.8173	0.0861	0.0021
20	0.0000	0.8173	0.0907	0.0021

BOTTOM WALL

TAP NO.	DU/DX	MACH NO.	D*	DD*
1	0.4216	0.8077	0.0155	-0.0009
2	-1.1125	0.8060	0.0223	-0.0004
3	-1.5025	0.7981	0.0290	-0.0000
4	-3.5534	0.7935	0.0353	0.0004
5	-7.0914	0.7721	0.0437	0.0025
6	-12.0694	0.7533	0.0508	0.0063
7	-3.1566	0.7367	0.0542	0.0084
8	8.0421	0.7459	0.0550	0.0075
9	7.1233	0.7563	0.0550	0.0059
10	6.3412	0.7634	0.0554	0.0046
11	-3.0033	0.7714	0.0567	0.0042
12	-8.1537	0.7563	0.0596	0.0058
13	0.6964	0.7520	0.0622	0.0063
14	8.0705	0.7742	0.0630	0.0025
15	3.0736	0.8001	0.0634	-0.0017
16	1.9875	0.8001	0.0664	-0.0037
17	1.6074	0.8149	0.0697	-0.0055
18	-2.0260	0.8121	0.0748	-0.0054
19	-1.6459	0.7998	0.0806	-0.0051
20	0.0000	0.7998	0.0861	-0.0046

UNIT REYNOLDS NO. = 357276.9 D* FPG = 0.0088

Figure 6.2.

WALL CP ERROR

TOP - 0.0072 BOTTOM - 0.0064

RESIDUAL ERRORS

X	U/UFS	V/UFS
-18.0000	0.0009	-0.0000
-17.0000	0.0009	-0.0000
-16.0000	0.0008	-0.0000
-15.0000	0.0007	-0.0000
-14.0000	0.0006	0.0000
-13.0000	0.0004	-0.0000
-12.0000	0.0003	-0.0000
-11.0000	0.0002	-0.0001
-10.0000	0.0001	-0.0001
-9.0000	-0.0000	-0.0001
-8.0000	-0.0002	-0.0001
-7.0000	-0.0004	-0.0001
-6.0000	-0.0005	-0.0000
-5.0000	-0.0005	-0.0000
-4.0000	-0.0002	0.0000
-3.0000	0.0005	-0.0000
-2.0000	0.0013	-0.0000
-1.0000	0.0021	-0.0001
0.0000	0.0024	-0.0001
1.0000	0.0021	-0.0001
2.0000	0.0014	-0.0001
3.0000	0.0008	-0.0001
~~4.0000~~	~~0.0005~~	~~-0.0001~~
5.0000	0.0003	-0.0001
6.0000	0.0002	-0.0001
7.0000	0.0001	-0.0001
8.0000	0.0000	-0.0001
9.0000	-0.0001	-0.0001
10.0000	-0.0001	-0.0000
11.0000	-0.0002	-0.0001
12.0000	-0.0001	-0.0001
13.0000	-0.0000	-0.0001
14.0000	0.0001	-0.0000
15.0000	0.0001	-0.0000
16.0000	0.0001	-0.0001
17.0000	0.0001	-0.0001
18.0000	0.0001	-0.0001

MODEL ERRORS			CP
-1.4979	0.0017	-0.0001	-0.0034
-0.7490	0.0022	-0.0001	-0.0044
0.0000	0.0024	-0.0001	-0.0048
0.7490	0.0022	-0.0002	-0.0045
1.4979	0.0018	-0.0001	-0.0036
2.2469	0.0013	-0.0001	-0.0025
2.9959	0.0008	-0.0001	-0.0016
3.7449	0.0005	-0.0001	-0.0010
4.4938	0.0004	-0.0001	-0.0007

EFFECT	DELTA CL
ALPHA ERROR = -0.0045 DEGREES	-0.0005
INDUCED CAMBER = 0.0007 DEGREES	-0.0013
VEL.ERROR CP = -0.0048	
AVERAGE = -0.0029	0.0048

```
########################
#  WALLS STREAMLINED   #
########################
```

Figure 6.3.

NPL SECTION ANALYSIS
9510

RUN NO. = 389

ALPHA = 3.00

UPPER SURFACE

%CHORD	CP LOCAL	CN LOCAL	CC LOCAL	CM LOCAL
0.0	1.0227	-0.0038	0.0079	0.0000
0.7	-0.7242	0.0036	-0.0062	-0.0001
1.0	-1.0257	0.0046	-0.0024	-0.0001
1.6	-1.2861	0.0064	-0.0027	-0.0002
2.0	-1.2979	0.0066	-0.0019	-0.0002
2.6	-1.3172	0.0070	-0.0018	-0.0002
3.1	-1.3824	0.0068	-0.0015	-0.0002
3.6	-1.3838	0.0146	-0.0026	-0.0006
5.2	-1.3318	0.0270	-0.0041	-0.0015
7.7	-1.2563	0.0314	-0.0038	-0.0025
10.2	-1.1720	0.0439	-0.0040	-0.0046
15.2	-1.1809	0.0591	-0.0044	-0.0091
20.2	-1.1601	0.0580	-0.0035	-0.0119
25.2	-1.1279	0.0563	-0.0028	-0.0143
30.2	-1.1249	0.0844	-0.0031	-0.0256
40.2	-1.0382	0.0779	-0.0021	-0.0314
45.2	-0.7661	0.0383	-0.0004	-0.0173
50.2	-0.5044	0.0189	-0.0001	-0.0095
52.7	-0.4348	0.0109	0.0000	-0.0057
55.2	-0.3941	0.0099	0.0001	-0.0054
57.7	-0.3548	0.0089	0.0001	-0.0051
60.2	-0.3606	0.0090	0.0002	-0.0054
62.7	-0.3504	0.0087	0.0003	-0.0055
65.2	-0.2664	0.0067	0.0003	-0.0043
67.7	-0.2778	0.0069	0.0003	-0.0047
70.2	-0.2736	0.0102	0.0007	-0.0071
75.2	-0.2361	0.0118	0.0010	-0.0088
80.2	-0.1737	0.0088	0.0010	-0.0070
85.3	-0.0976	0.0049	0.0007	-0.0041
90.1	-0.0143	0.0007	0.0001	-0.0006
95.2	0.0626	-0.0031	-0.0007	0.0029
****	0.1394	-0.0034	0.0003	0.0034

LOWER SURFACE

%CHORD	CP LOCAL	CN LOCAL	CC LOCAL	CM LOCAL
0.0	1.0684	0.0053	0.0098	0.0000
1.0	0.8331	0.0074	0.0098	-0.0003
1.8	0.5949	0.0127	0.0059	-0.0004
5.3	0.1998	0.0085	0.0026	-0.0005
10.2	-0.0488	-0.0024	-0.0005	0.0003
15.3	-0.1493	-0.0074	-0.0009	0.0012
20.2	-0.2291	-0.0142	-0.0009	0.0029
27.7	-0.3163	-0.0275	-0.0000	0.0076
37.6	-0.3310	-0.0288	0.0017	0.0107
45.1	-0.2158	-0.0162	0.0020	0.0072
52.6	-0.0488	-0.0037	0.0006	0.0019
60.1	0.0888	0.0067	-0.0011	-0.0040
67.5	0.3073	0.0179	-0.0030	-0.0121
71.7	0.3629	0.0318	-0.0039	-0.0228
85.1	0.4490	0.0412	-0.0036	-0.0351
90.1	0.4430	0.0221	-0.0000	-0.0200
95.1	0.3825	0.0190	0.0005	-0.0180
****	0.1311	0.0032	0.0003	-0.0032

	PRESSURE	SUCTION	TOTAL
CN	0.6319	0.0755	0.7075
CC	-0.0350	0.0193	-0.0157
CM	-0.1870	-0.0844	-0.2715

WING PERFORMANCE

CL	CD	CM
0.7073	0.0214	-0.2715

Figure 6.4.

TRANSDUCER OUTPUT
CP VALUES CHANNELS 2-4

	1	2	3	4
1	26.5666	-0.0463	0.0891	-0.0107
2	3.6152	-0.0063	0.1375	-0.0069
3	42.8183	0.0038	0.1788	0.0075
4	49.5844	-0.0171	0.1154	0.0176
5	~~55.2891~~	~~0.0006~~	~~0.1331~~	~~-0.0649~~
6	55.5544	-0.0248	0.1037	0.1066
7	26.5003	-0.0777	0.1504	0.1431
8	55.9856	-0.2113	0.1702	0.1230
9	57.4449	-0.4048	0.1594	0.1015
10	57.4781	-0.5595	0.8655	0.0858
11	56.3836	-0.3903	0.0904	0.0680
12	54.6921	-0.2219	0.0778	0.1014
13	26.5334	-0.1621	0.1490	0.1108
14	52.8015	-0.0915	0.1700	0.0617
15	53.0005	-0.0229	0.1365	0.0120
16	52.5362	-0.0197	-0.2664	0.0120
17	52.1382	-0.0164	-0.2778	-0.0207
18	52.0719	-0.0227	-0.2736	-0.0144
19	26.6993	-0.0259	-0.2361	0.0125
20	50.1150	0.0341	-0.1737	0.1953
21	43.9792	1.0152	-0.0976	-0.0094
22	38.0755	0.0032	-0.0143	0.0507
23	36.9146	-0.0306	1.1970	0.1238
24	35.9860	-0.1368	0.1394	0.1330
25	26.9978	1.0057	0.0876	0.0514
26	35.0905	-0.2630	0.1088	0.1102
27	35.2231	-0.5160	0.1518	0.0619
28	34.9910	-0.7459	0.1370	1.0440
29	2.5870	-0.2625	0.1574	0.0932
30	7.8605	-0.2378	0.1383	0.0945
31	26.5334	-0.2352	0.1550	0.0680
32	13.2004	-0.3198	0.1407	0.0623
33	22.0559	-0.1557	0.1401	0.1152
34	27.6279	-0.1036	0.1227	0.0365
35	29.9165	-0.0356	0.1471	0.0107
36	31.7075	-0.0254	0.1513	0.1195
37	26.5666	-0.0146	0.1070	-0.0069
38	33.6643	-0.0140	0.3073	-0.0075
39	33.9960	-0.0317	0.3629	0.0000
40	31.4090	0.0343	0.4490	-0.0289
41	27.6279	-0.0153	0.4430	0.0964
42	24.5434	-0.2015	0.3825	0.1291
43	26.5334	-0.6707	0.1311	0.0768
44	11.3099	-0.2734	0.1413	0.0611
45	1.6252	0.0057	0.1131	0.0504
46	9.7842	-0.0922	0.0778	0.1121
47	1.6252	-0.0165	0.0892	0.0787
48	0.2653	-0.0540	-0.0742	0.0964

Figure 6.5.

RUN 389 OUTPUT

	EXT VEL.		MOVEMENT		Y CO-ORD	
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6.0000	0.0117	0.0013	0.0007	-0.0005	0.0154	0.0353
9.0000	0.0008	0.0011	0.0005	-0.0003	0.0179	0.0471
12.0000	-0.0007	-0.0021	-0.0001	0.0000	0.0546	0.0881
15.0000	0.0048	-0.0056	-0.0000	0.0004	0.0867	0.1115
18.0000	0.0018	-0.0170	-0.0003	0.0012	0.1444	0.1544
20.0000	0.0119	-0.0279	-0.0004	0.0008	0.2857	0.1460
21.0000	0.0255	-0.0408	0.0004	-0.0000	0.3369	0.1057
22.0000	0.0602	-0.0386	0.0014	-0.0014	0.4937	0.0711
23.0000	0.1059	-0.0394	0.0029	-0.0030	0.5257	-0.0047
24.0000	0.1359	-0.0456	0.0041	-0.0039	0.5652	-0.0082
25.0000	0.1034	-0.0247	0.0034	-0.0015	0.5882	-0.0118
26.0000	0.0681	-0.0262	0.0014	0.0000	0.4967	-0.0083
27.0000	0.0506	-0.0301	0.0006	0.0005	0.4577	0.0461
29.0000	0.0267	-0.0189	0.0006	0.0004	0.3369	0.0532
32.0000	0.0067	-0.0033	0.0005	0.0011	0.2595	0.0637
35.0000	0.0069	-0.0025	0.0001	0.0015	0.2430	0.0118
38.0000	0.0042	0.0057	0.0004	0.0015	0.2060	0.0461
41.0000	0.0071	0.0045	0.0002	0.0016	0.1725	0.0200
44.0000	0.0070	-0.0039	0.0007	0.0021	0.1389	-0.0130
47.0000	0.0070	-0.0039	0.0007	0.0023	0.0863	0.0439
50.0000	0.0074	-0.0023	0.0000	0.0000	0.0000	0.0000
53.0000	0.0074	-0.0023	0.0000	0.0000	0.0000	0.0000

Figure 6.6

RUN 389 DATA

TOP WALL

JACK	MACH NO.					

MACH NO. RECORD (1-20) = 20

JACK	MACH NO.					
1		0.0158	0.7634	0.0007	467	467
2		0.0183	0.7714	0.0005	566	566
3		0.0546	0.7563	-0.0001	551	550
4		0.0875	0.7520	-0.0000	611	610
5		0.1456	0.7742	-0.0003	639	638
	0.8101					
6		0.2899	0.8001	-0.0004	692	691
	0.8256					
7		0.3445	0.8001	0.0004	746	746
	0.8748					
8		0.5067	0.8149	0.0014	890	891
	0.9339					
9		0.5445	0.8121	0.0029	922	924
	1.0569					
10		0.5870	0.7998	0.0041	941	944
	1.1771					
11		0.6070	0.7998	0.0034	933	935
	0.9315					
12		0.5097	0.9032	0.0014	896	897
	0.9200					
13		0.4665	0.8758	0.0006	869	869
	0.9189					
14		0.3407	0.8436	0.0006	870	870
15		0.2582	0.8159	0.0005	775	775
16		0.2401	0.8144	0.0001	318	318
17		0.2027	0.8130	0.0004	362	362
18		0.1696	0.8159	0.0002	339	339
19		0.1368	0.8173	0.0007	300	300
20		0.0842	0.8173	0.0007	365	365

BOTTOM WALL

1	0.0345	0.8077	-0.0005	583	583
2	0.0467	0.8060	-0.0003	502	502
3	0.0881	0.7981	0.0000	523	522
4	0.1119	0.7935	0.0004	531	530
5	0.1569	0.7721	0.0012	495	493
6	0.1523	0.7533	0.0008	382	381
7	0.1141	0.7367	-0.0000	474	474
8	0.0786	0.7459	-0.0014	509	510
9	0.0012	0.7563	-0.0030	568	570
10	-0.0035	0.7634	-0.0039	617	620
11	-0.0076	0.7714	-0.0015	576	577
12	-0.0025	0.7563	0.0000	593	592
13	0.0524	0.7520	0.0005	543	542
14	0.0557	0.7742	0.0004	662	661
15	0.0620	0.8001	0.0011	649	648
16	0.0080	0.8001	0.0015	320	318
17	0.0407	0.8149	0.0015	242	240
18	0.0146	0.8121	0.0016	210	208
19	0.0180	0.7998	0.0021	211	209
20	0.0393	0.7998	0.0023	295	293

WALL & MODEL OUTPUT RECORD NO. = 21

==

STOP --

Figure 6.7.

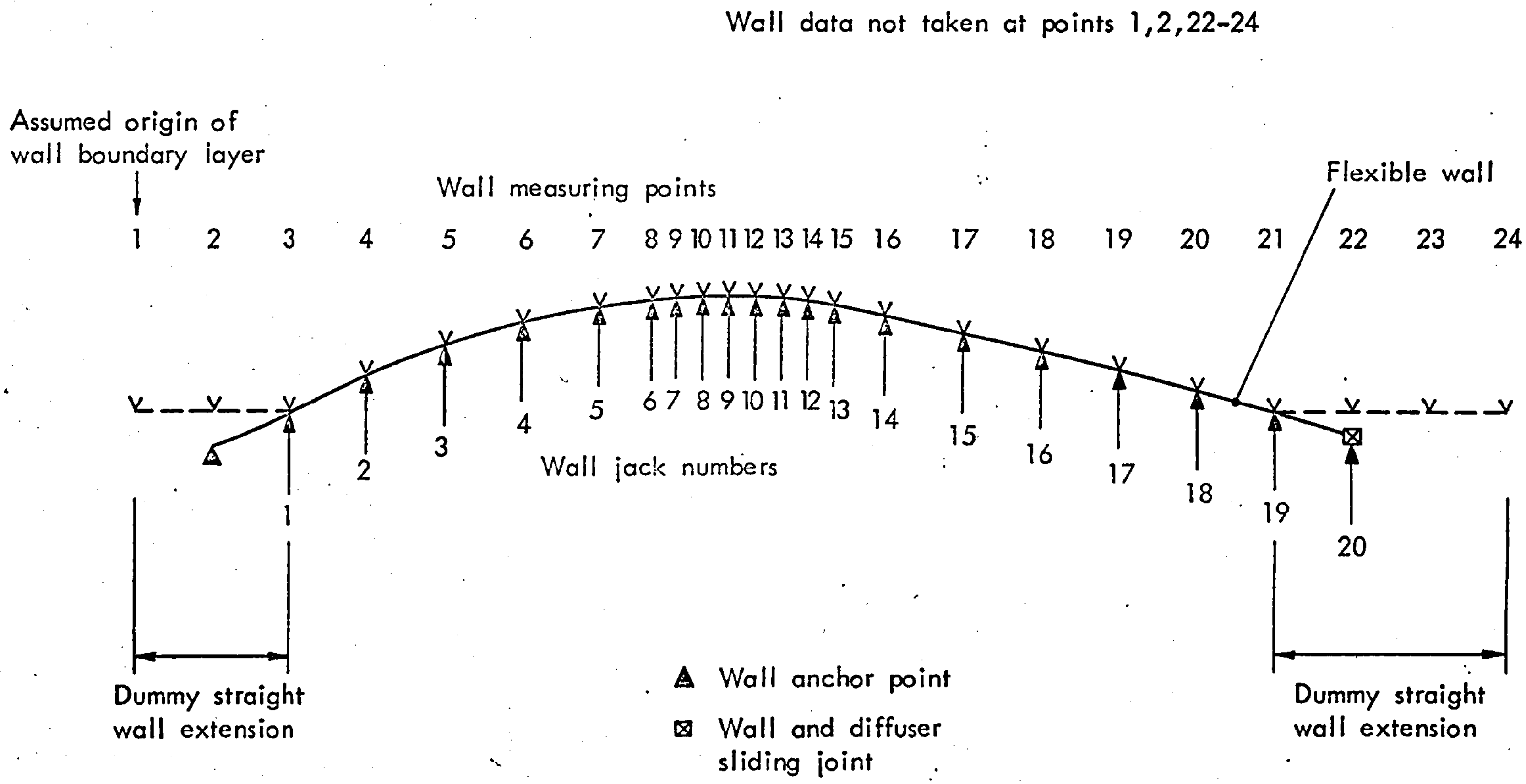


FIG. 7 SOFTWARE REPRESENTATION OF EACH FLEXIBLE WALL

APPENDIX A. DIGITAL INPUT/OUTPUT PROTOCOL

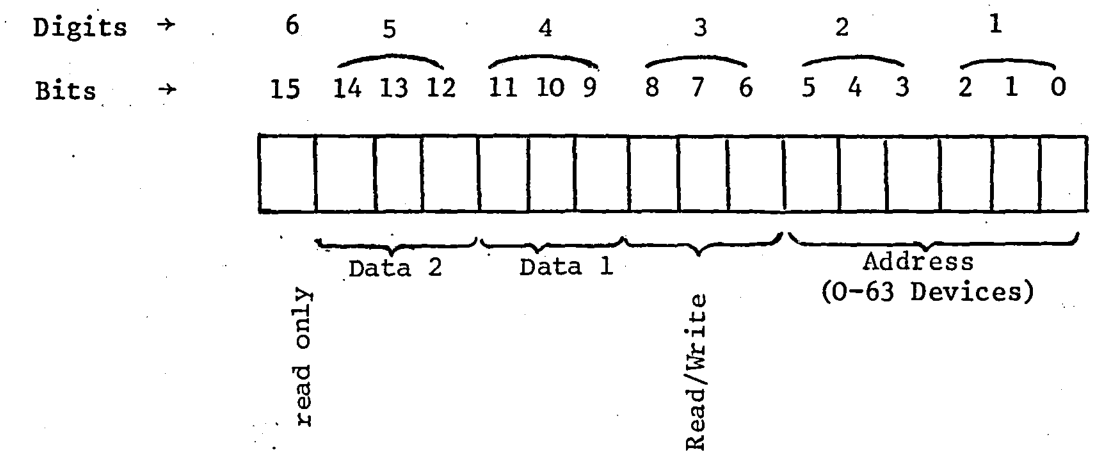
FOR TSWT CONTROL SYSTEM

The transfer of digital information between the computer and wind tunnel hardware is an important part of the on-line control system. Digital I/O involves a complex interaction of system software and hardware. A command code has been devised to simplify the operation of the control system. The protocol of digital I/O are described in the following sections.

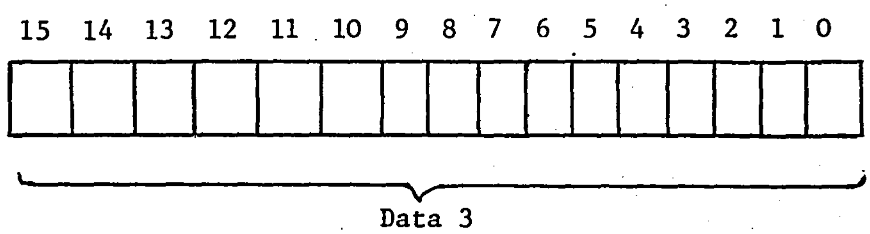
Data Format

Each packet of information sent to the wind tunnel consists of the data destination and operation plus the data itself. Information from the wind tunnel consists of data only. All sets of information are contained in a single 16-bit word in the following manner:-

Digital Output Word (Equivalent to a six digit octal number in binary code)



Digital Input Word



Device Addresses

Each address consists of six bits of binary information which corresponds to a decimal number referred to as the software address. The following scheme has been chosen:

Device		Software Address	Binary Representation
			5 4 3 2 1 0 — 5 4 3 2 1 0
Stepper Motors		1 → 44	0 0 0 0 0 1 - 1 0 1 1 0 0
			5 4 3 2 1 0
Pulse Sequence Generator		45	1 0 1 1 0 1
Scanivalves	(1)	46	1 0 1 1 1 0 - 1 1 0 0 0 0
	(2)	48	
Encoders	(1)	47	1 0 1 1 1 1 - 1 1 0 0 0 1
	(2)	49	
			5 4 3 2 1 0 — 5 4 3 2 1 0
Power Supplies etc.		50 → 63	1 1 0 0 1 0 - 1 1 1 1 1 1

Data Operation

The transfer of data takes three forms - read only, write only and write before read. These operations determine the type of data to be sent, if any. For example, the read only function requires no data from the computer other than the device address.

Information on data operation is sent to the wind tunnel by adding a chosen software value to the decimal equivalent of the digital ouput word. This effectively sets the required bits for correct information transfer. The software values are chosen thus:

Data Information	Binary Representation: 15	Binary Representation: 8 7 6	Software Value
Read only	1	0 1 0	-32768 + 128 = -32640
Write only	0	0 0 0	0 + 0 = 0
Write before read	0	0 1 0	0 + 128 = 128
Digit	6	3	

Data

Actions at the wind tunnel are determined by the corresponding data 1 or 2 bits of the output word. Data 3 bits of the input word allow checks to be made on system devices.

The output data is transferred by the setting of data 1 and 2 bits, achieved by the modification of the digital output word as for the data operating information.

The software values were chosen thus:

Function	Binary Representation	Software Value
	Data 1	
a) Motor direction	11 10 9	
Stop	0 0 0	0
Forward-go	1 0 1	2560
Reverse-go	1 1 0	3072
Digit	4	
	Data 2	
b) Scanivalve Move	14 13 12	
-Home	1 0 1	20480
-Step on one	1 0 0	16384
Digit	3	
c) Pulse sequence generator	14 13 12	
-Start	0 0 1	4096
d) Motor Power Supply	14 13 12	
-Off	0 0 0	0
-On	0 1 0	8192
Digit	5	

Function	Binary Representation	Software Value
e) Pulse sequence generator Increment step size (currently non operational). No.	14 13 12 11 10 9 8 $\underbrace{\boxed{}\boxed{}}_{10s}\underbrace{\boxed{}\boxed{}\boxed{}}_{100s}\underbrace{\boxed{}\boxed{}}_{1000s}$	0 → 32512

Input data will be in binary code to allow simple software manipulation. The complete input word will always be read regardless of the quantity of information being transferred.

The types of data are as follows:

Device	No. of bits of Information
Scanivalve - encoder output	7
- at home position	1
Pulse sequence generator	
- finish pulse	1
- step value	7
System monitor	16
Motor direction	3

Command Coding

Each command sent to the wind tunnel must be unambiguous and provide information on data destination and operation and the data itself. This is achieved by placing a decimal code number in binary on the 16 output lines. This command number N is then decoded by the wind tunnel hardware and some operation performed.

The command number for each operation is determined by the code

$$N = \boxed{\text{Device Address}} + \boxed{\text{Data Operation Value}} + \boxed{\text{Data Value}}$$

using software values only. For example, if the scanivalve (1) is required to step on one port, the following command number would be sent to the wind tunnel

$$N = 46 + 0 + 16384 = \underline{16430}$$

<u>Summary</u>

The interface between wind tunnel hardware and the computer is by means of 16 output lines and 16 input lines plus control lines. The voltage levels on the output lines are controlled by the software generated command numbers described above. The voltage levels on the input lines are controlled by software selected tunnel hardware. All transfer of data is accompanied by a handshaking procedure to ensure correct sequencing of communication operations.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-165941		

4. Title and Subtitle	5. Report Date
Control Software for Two Dimensional Airfoil Tests Using a Self-Streamlining Flexible Walled Transonic Test Section	July 1982
	6. Performing Organization Code
7. Author(s) S.W.D. Wolf	8. Performing Organization Report No.
9. Performing Organization Name and Address University of Southampton, Department of Aeronautics and Astronautics, Southampton, England	10. Work Unit No.
	11. Contract or Grant No. NSG-7172
	13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration, Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

This is a semi-annual progress report for the priod to February 1982, on work undertaken on NASA Grant NSG-7172 entitled "The Self Streamlining of the Test Section of a Transonic Wind Tunnel." The Principal Investigator is Dr. M. J. Goodyer.

16. Abstract

The current operation of the Transonic Self-Streamlining Wind Tunnel (TSWT) involves on-line data acquisition with automatic wall adjustment. A tunnel run consists of streamlining the walls from known starting contours in iterative steps and acquiring model data. Each run performs what is described as a streamlining cycle. The associated control software is presented here.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Facilities, research and support Flexible wall wind tunnel	Unclassified - Unlimited Star Category - 09

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	92	A05

N-305

For sale by the National Technical Information Service, Springfield, Virginia 22161

End of Document